UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

APLICACIÓN DE LA METODOLOGÍA SCRUM PARA LA ENSEÑANZA DE SOLUCIÓN ALGORÍTMICA DE PROBLEMAS CON LA TECNOLOGÍA PLAYGROUNDS MEDIANTE EL LENGUAJE DE PROGRAMACIÓN SWIFT

T E S I S

PARA OBTENER EL TÍTULO DE:

LICENCIADA EN MATEMÁTICAS APLICADAS Y COMPUTACIÓN

P R E S E N T A:

PAOLA VANESSA OROZCO DEL ANGEL

DIRECTOR DE TESIS:

PH. EDUARDO ELOY LOZA PACHECO

SANTA CRUZ ACATLÁN, EDO. DE MÉXICO, 2020

Contenido

Resumen

En el presente trabajo de tesis se expone una solución a la problemática Universitaria donde se requiere el aprendizaje de nuevas e innovadoras herramientas tecnológicas como los **Playgrounds**, mediante el lenguaje de programación **Swift**. La solución propuesta es la creación, desarrollo y mejora iterativa de un club de programación aplicando la metodología *Scrum* para lograr un proceso estructurado, que mida e incremente la efectividad del uso de los recursos materiales, académicos, científicos y tecnológicos.

El trabajo inicia con la descripción del problema, alineado y delimitado a una solución propuesta, el planteamiento de un objetivo general y específico. Seguido de un contexto necesario acerca de las metodologías ágiles con el fin de describir las técnicas de ***Scrum***. Dicha estrategia de trabajo es importante pues su objetivo, propósito, teorías, principios, aspectos, fases y procesos son relevantes en el desarrollo efectivo del proyecto del club de programación que se implementó en el ***iOS Development Lab*** de la **Facultad de Estudios Superiores Acatlán (F.E.S. Acatlán)**. De igual manera, se mostrará la importancia de la **Solución Algorítmica de Problemas (SAP)**, ya que es una de las bases fundamentales para la programación dentro de las ciencias de la computación, que es el punto central dentro de las enseñanzas del club de programación. Se revisan y describen las tecnologías de Apple como lo son Xcode, Playgrounds y el lenguaje de programación Swift. Así mismo, se explica el punto de convergencia entre los temas tecnológicos para el beneficio educativo de los estudiantes. En donde se muestra la unión de *Scrum* y SAP que se utiliza para aplicar la creación de la solución propuesta y resolver la problemática planteada. El trabajo muestra el desarrollo de una guía de aprendizaje de programación que se aplicó en el club. Finalmente, se presentan los resultados, hallazgos obtenidos en las generaciones 2019-2 y 2020-1, mejoras a futuro y las conclusiones del trabajo.

Lista de Imágenes

Lista de Tablas

Lista de Gráficas

Lista de Ejemplos

Lista de Fotografías

Capítulo I. Presentación

1.1 Introducción

La educación tecnológica es una necesidad básica en el siglo XXI para darle solución a las problemáticas del mundo real, ya que todos los artefactos tecnológicos que surgen a diario se vuelven cada vez más complejos. Los profesionales de las ciencias de la computación de igual manera tienen como necesidad aprender y dominar dichas tecnologías, de ahí diversas empresas tecnológicas como Google, Microsoft Research, la Free Software Foundation y Apple realizan iniciativas para poder extender el uso de éstas. En el caso de **Apple**, ha trabajado desde el año de 1984 confiando en la visión de un entorno educativo que se pueda mejorar y redefinir gracias a la tecnología, ampliando las posibilidades de aprendizaje, por lo que crearon centros llamados: "*Apple Distinguished Schools*". Centros de liderazgo y excelencia educativa que ponen en práctica esta visión, haciendo a las instituciones las más innovadoras del mundo. [1]

En México se cuenta con laboratorios llamados ***iOS Development Lab***, los cuales fueron creados como una propuesta de desarrollo educativo entre Apple y las mejores instituciones educativas del país. En el caso de la **Universidad Nacional Autónoma de México**, existen tres: dos en **Ciudad Universitaria** (el primero en la Facultad de Contaduría y Administración y el segundo en la Facultad de Ingeniería); y uno ubicado en la **F.E.S. Acatlán**. El objetivo de estos laboratorios es combinar el aprendizaje de una tecnología de cómputo como Xcode y el lenguaje de programación Swift, para incentivar a toda la comunidad universitaria a la creación de aplicaciones móviles, que tengan fines educativos, sociales y hasta comerciales para el mundo. Una vez terminadas, puedan publicarse en la App Store, teniendo así, alumnos competitivos dentro de las ramas tecnológicas. [10]

Tomando como ejemplo a seguir a estos centros de liderazgo (*Apple Distinguished Schools*), se pretendió llevar esta visión a los ***iOS Development Lab*** en la UNAM, usando como pilares esenciales: el aprendizaje, la enseñanza, el liderazgo y el entorno. Creando oportunidades donde los estudiantes optimicen el aprendizaje cuando estén plenamente comprometidos, contando con los recursos adecuados y confiando en que su trabajo es importante. [2]

En el presente trabajo se expone el proceso de desarrollo de un proyecto educativo que mantuvo como metas: **1)** Prevalecer los recursos y visión de los ***iOS Development Labs***, **2)** Incrementar el desarrollo institucional UNAM (Plan 2019-2023) [11], a través del aprendizaje de herramientas didácticas, tecnologías de Apple como los Playgrounds, y de Técnicas de Ingeniería de Software y de Ciencias

de la Computación como lo es la SAP. Para lograr un proceso óptimo y estructurado se decidió que fuera aplicado a una de las metodologías ágiles más importantes en la iniciativa privada: *Scrum*.

El proyecto educativo consistió en la creación de un **club de programación**, que fue impartido en el **Centro de Desarrollo de Aplicaciones Móviles** - ***iOS Development Lab*** (C.E.D.A.M.) de la F.E.S. Acatlán. A continuación, en la *Fotografía I* se muestra el *iOS Development Lab*:

Huerta, E. (2019). Fotografía iOS Development Lab. [Fotografía 1]

En la investigación se comenzará por la definición, características, principios y forma de trabajo de la metodología *Scrum*, con la que se llevó a cabo la realización del proyecto, siguiendo la descripción de lo que es la comprensión de problemas mediante la modularización, diagramas de flujo, análisis y diseño de algoritmos. Posteriormente se presenta una explicación sobre la tecnología de los Playgrounds, describiendo lo que son, su aplicación, prestaciones multimedia, y funcionalidades con respecto a la enseñanza didáctica. Acompañado de la implementación del lenguaje de programación Swift, completando de esta manera la comprensión de soluciones a problemas en los paradigmas de la programación. Más adelante, se mostrará la implementación de Scrum para la planeación, desarrollo, mantenimiento y mejoramiento **iterativo** del club de programación. Seguiremos con los resultados obtenidos al término del periodo de la implementación del proyecto, resaltando las condiciones finales para el seguimiento efectivo del mismo. Finalmente, se dará una conclusión de lo que fue participar en un proyecto de tan gran impacto en la comunidad universitaria y los aprendizajes que se obtuvieron a su término.

1.2 Objetivo general

Aplicar la metodología ágil: *Scrum*, para la creación y mejora iterativa del proceso de enseñanza en la solución algorítmica de problemas a través de Playgrounds con el uso del lenguaje de programación Swift con el apoyo de las tecnologías del laboratorio *iOS Development Lab* a los alumnos de la F.E.S. Acatlán mediante un club de programación.

1.3 Objetivos particulares

- Revisar de las metodologías ágiles.
- Describir de la metodología *Scrum,* así como sus principios, aspectos y procesos.
- Exponer de los tipos de representación y razonamiento utilizando la solución algorítmica de problemas.
- Analizar las tecnologías Apple en la educación.
- Aplicar las fases de *Scrum* al proyecto educativo.
- Creación y mejora del proceso de enseñanza.

1.4 Hipótesis

Es posible mejorar la calidad de aprendizaje en la enseñanza a la programación enfocada a la solución algorítmica de problemas, a través de formas didácticas como lo son hoy en día las herramientas tecnológicas que nos ofrecen los entornos de desarrollo de Apple, utilizando una de las metodologías ágiles como lo es Scrum para la planeación, desarrollo, mantenimiento y mejora iterativa de programas escolares como un club de programación.

1.5 Justificación

El siguiente trabajo tiene como objetivos: 1) Expandir las habilidades tecnológicas y la experiencia de un club de programación, PWR CODE, a más personas, 2) Aprender de cada iteración para elevar los estándares de aprendizaje y 3) Mantener para las generaciones futuras un modelo de trabajo que siempre se mantenga a la vanguardia. La primera parte de este trabajo presenta una investigación necesaria sobre las Metodologías Ágiles en el ámbito de la Ingeniería de Software, la SAP situada en el campo de las matemáticas computacionales, las nuevas tecnologías de computo de Apple como Playgrounds, XCODE, Swift, etcétera; y la metodología *Scrum,* para llegar al entendimiento común de las soluciones propuestas para el club de programación **PWR CODE** impartido en las instalaciones del laboratorio *iOS*

Development Lab de la F.E.S. Acatlán. La segunda parte del trabajo muestra la creación de un esquema de trabajo basado en lo aprendido en todas las generaciones de PWR CODE y en la experiencia (*expertise)* obtenida.

El club de programación, PWR CODE, ha logrado obtener el interés dentro de la comunidad universitaria a lo largo de su corta existencia, dejando sobre la mesa nuevos retos en la calidad de la educación y en el proceso de trabajo para las personas encargadas de dicho proyecto educativo con el fin de poner a la universidad a la vanguardia. Esta tesis da un modelo de trabajo estándar y moldeable a las necesidades del club de programación ya identificadas y futuras, apoyándose de la metodología *Scrum* mayormente utilizada en la industria del software. ***Scrum*** fue la metodología selecciona porque su alcance es tan grande que se puede adaptar para cualquier tipo de sector y además se caracteriza por la colaboración y comunicación conjunta que deben tener los integrantes del proyecto para lograr un producto exitoso. [6] Por otro lado, esta tesis también aporta mejoras en la calidad del proceso educativo proponiendo nuevo material de trabajo, actividades y temarios que complementen y apoyen el aprendizaje de los estudiantes como la integración de la SAP y el uso de tecnologías innovadoras como los Playgrounds de Apple. La **SAP** fue seleccionada como una nueva e indispensable aportación al club, ya que da un gran valor durante el proceso de desarrollo lógico-matemático en los estudiantes para la resolución de problemas computacionales, mientras que los **Playgrounds** fueron una herramienta tecnológica seleccionada debido a que apoyan a los estudiantes a crear la representación de una o varias soluciones encontradas en el proceso de SAP en forma de código mediante el lenguaje de programación **Swift**.

En la *Fotografía 2* se puede observar a dos estudiantes de PWR GRL CODE teniendo una plática sobre el lenguaje de programación Swift.

Huerta, E. (2019). Fotografía Alumnas del club PWR GRL CODE. [Fotografía 2]

1.6 Contenido estructural de la investigación

Capítulo I. Presentación

En este capítulo se muestra la introducción, justificación, objetivo general y objetivos particulares, así como la estructura del trabajo.

Capítulo II. Metodologías Ágiles

El capítulo II describe los puntos esenciales de una metodología ágil y sus principales fundamentos dentro de los proyectos y relevancia.

Capítulo III. Scrum

En este capítulo se presentará la teoría y técnicas de la Metodología *Scrum* con la que se llevó a cabo la realización del proyecto: “club de programación”.

Capítulo IV. Solución Algorítmica de Problemas

Se describen los conceptos básicos de lo que conlleva la formulación y comprensión de problemas, utilizando diferentes tipos de representación, razonamiento y herramientas cognitivas.

Capítulo V. Uso de Playgrounds con Swift

Para continuar, en este capítulo se describe la herramienta tecnológica llamada Playgrounds y su implementación con el lenguaje de programación Swift, que ofrece el entorno de desarrollo Xcode creado por Apple y el cómo puede llegar a ser una técnica didáctica de aprendizaje.

Capítulo VI. Desarrollo del proyecto aplicado a las fases de Scrum

Una vez explicadas las herramientas tecnológicas y técnicas que se utilizaron para el desarrollo del proyecto educativo, se forma el capítulo VI, describiendo la creación, desarrollo, mejora iterativa y mantenimiento del club de programación con el apoyo del marco teórico *Scrum*.

Capítulo VII. Resultados del proyecto (2019-2/2020-1) y nuevas propuestas

El capítulo VII describe los resultados obtenidos al finalizar el período de implementación del proyecto 2019-2 y 2020-1.

Conclusiones

Finalmente, se presenta una reseña clara de lo aprendido después de las investigaciones realizadas y descritas en los capítulos anteriores, una reflexión sobre la hipótesis y los resultados obtenidos.

Capítulo II. Metodologías Ágiles

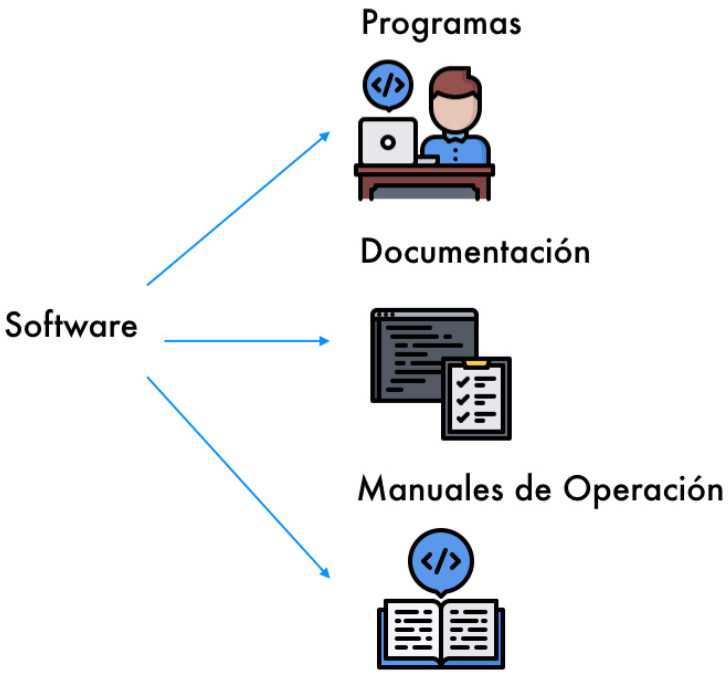
2.1 Ingeniería de Software

La **Ingeniería de Software** se concentra en la sistematización de procesos científicos bien definidos para desarrollar productos de software de alta calidad. Las actividades de Ingeniería de Software esperan producir documentos adecuados después de cada etapa de desarrollo; El desarrollo debe llevarse a cabo según normas y directrices bien documentadas dentro de la **metodología a desarrollar**, así como completarse a tiempo y dentro del presupuesto.

Es importante tener bien definidos los conceptos de "**Software**" y "**Programa**" ya que forman parte fundamental del entendimiento de lo que es la Ingeniería de Software. Un **programa** es un conjunto de instrucciones que se escriben para un propósito en específico, puede contener comentarios entre líneas que ayuden al entendimiento del programa. Por otro lado, el **software** es una combinación de uno o varios programas, documentación (producida durante el desarrollo) y manuales de operación (entregados con el programa al cliente en el momento del lanzamiento del producto).
Por lo tanto, un proyecto de software de cualquier tamaño y funcionalidades siempre debe de estar asociado a una documentación y manuales de operación. [12]

A continuación, en la *Imagen A* se puede observar mejor lo ya mencionado:

Imagen A: Elementos del Software



2.2 Concepto de Metodología

Una **metodología** se define como el conjunto de procedimientos y técnicas de rigor científico, que se aplican sistemáticamente durante un proceso de desarrollo de un proyecto, para alcanzar un resultado teóricamente válido o bien que satisfaga las necesidades del cliente. [6]
Es decir, es un proceso administrativo de planeación, dirección, organización y control; en este sentido logramos ver que la metodología da un soporte fundamental y es quien define una parte del éxito del proyecto.

La metodología se divide en 5 elementos básicos:

1. Fases: Se dividirán las tareas a realizar en el proyecto.

2. Métodos: Se identificarán los mecanismos con los que se realizará el desarrollo del producto, descomponiendo las fases en tareas más pequeñas.

3. Técnicas y herramientas: Indicarán cómo se debería de resolver cada tarea y que herramientas podrían usarse.

4. Documentación: Se definirá la documentación a entregar durante el proyecto, esto servirá para recoger los resultados.

5. Control y evaluación: Se deben de realizar a lo largo de todo el ciclo de vida del proyecto. Consistirá en aceptar, rechazar, comprobar y replantear todos los resultados que se vayan obteniendo en el proyecto, para no perder los objetivos iniciales. [6]

Existe una clasificación de las metodologías, en este caso, se estudiarán aquellas con grado de formalismo:

- Metodologías Pesadas
  Son las metodologías tradicionales, los métodos de trabajo son muy formales, conllevan a realizar una gran carga de trabajo de gestión y generar una gran cantidad de documentación.

- Metodologías Ágiles
  Son las últimas en aparecer y se basan en dar respuestas a los problemas con los que se encuentran las metodologías tradicionales. Usan el concepto de **adaptación** a los requisitos que no se conocen, en lugar de la predicción. Sólo genera la documentación necesaria. [6]

2.3 Métodos Ágiles

Cada metodología ágil individual enfoca los valores y los principios de una manera ligeramente diferente. A continuación, se presentarán las metodologías ágiles que tuvieron más auge en su desarrollo:

- **Lean Kanban**
  Es una técnica creada por Toyota, “Kanban” es una palabra japonesa que hace referencia a “tarjetas visuales”, y se utiliza para controlar el avance de trabajo. Su principal objetivo es la reducción de tiempo en cada iteración. Mediante la detección de retrasos ayuda a detectar errores en las primeras etapas en la producción, aumentando la productividad y reduciendo en consecuencia la cantidad total de trabajo necesario para finalizar una tarea. [6]

- **Programación Extrema** (*Extreme Programming*)
  Creada en Chrysler Corporation es implementada mayormente para proyectos con requisitos imprecisos y cambiantes enfocándose en evitar el aumento radical del costo del producto con el paso del tiempo, haciendo uso de un equipo de desarrollo con una formación y capacidad de aprendizaje elevada. [6]

- ✓ **Métodos Crystal** (*Crystal Methods*)
  Introducida por Alistair Cockburn, su principal intención es centrarse en las personas, ser ligeros y fáciles de adaptar, es por lo que se divide por tamaño y complejidad, haciendo referencia a que deben de existir políticas diferentes para equipos diferentes, como se observa a continuación:

  - Claro: 6 o menos personas en el equipo.
  - Amarillo: 7 a 20 personas en el equipo.
  - Naranja: 21 a 40 personas en el equipo.
  - Rojo: 41 a 80 personas en el equipo.
  - Marrón: 81 a 200 personas en el equipo.

  En estos métodos se contemplan 4 roles, que son:

  1. Patrocinador Ejecutivo (*Executive Sponsor*).
  2. Diseñador Líder (*Lead designer*).
  3. Desarrolladores.
  4. Usuarios experimentados. [6]

- **Métodos de Desarrollo de Sistemas Dinámicos** (D*ynamic Systems Development Methods*)
  Por sus siglas en inglés DSMS, estos métodos se centran en la calidad y el esfuerzo en términos de costo y tiempo usando la priorización de los entregables con la técnica MoSCoW: Debe tener (*Must Have*), Debería tener (*Should Have*), Podría Tener (*Could Have*) y No tendrá (*Won't Have*).
  Consta de 6 fases distintas:

  1. Pre-Proyecto.
  2. Viabilidad.
  3. Fundamentos.
  4. Exploración e Ingeniería.
  5. Desplazamiento.
  6. Evaluación de Beneficios. [7]

- **Desarrollo Orientado a Funcionalidades** (F*eature Driven Development*)
  Por sus siglas en inglés FDD, fue creado por Jeff De Luca, su objetivo principal es concluir un proyecto mediante su fragmentación en pequeñas funcionalidades valoradas por el cliente. Donde se entregan aproximadamente en un periodo de dos semanas. Este desarrollo responde a dos principios: el desarrollo de software es una actividad humana y es una funcionalidad valorada por el cliente. [7]

- **Desarrollo Guiado por Pruebas** (*Test Driven Development*)
  Este desarrollo fue introducido por Kent Beck, es un método de desarrollo de software que implica pruebas de código, si está bien implementado, se continúa, de lo contrario se configura hasta que pase las pruebas, esto por consecuencia agiliza el proceso de código limpio. [7]

- **Desarrollo Adaptativo de Software** (*Adaptive Software Development*)
  Por sus siglas en inglés ASD, fue integrado por Jim Highsmith y Sam Bayer, es un proceso de desarrollo con una constante adaptación a los procesos de trabajo dentro de los grandes proyectos, se fomenta el desarrollo iterativo e incremental con el uso de prototipos. [7]

- **Proceso Unificado Ágil** (A*gile Unified Process*)
  Desarrollado por Scott Ambler, es la evolución de IBM's Rational Unified Process, considera la combinación de metodologías ágiles y las tradicionales con el objetivo de brindar un producto funcional de la mejor calidad. [7]

- **Desarrollo Guiado por el Dominio** (*Domain Driven Development*)
  Conceptualizado por Eric Evans, fue creado para administrar diseños complejos y evolutivos, en donde se maneja como núcleo las ideas del negocio convirtiéndolos en artefactos de software. [7]

- **Metodología Scrum**
  Su principal principio es lograr la simplicidad y escalabilidad, a través de miembros del equipo que trabajen juntos y de forma eficiente obteniendo productos complejos y sofisticados. [7]

Como se observa anteriormente, cada una de las metodologías descritas arriba tienen como objetivo eficientar y mejorar el desarrollo de la entrega del producto. Para este trabajo se seleccionó la metodología Scrum.

2.4 Manifiesto Ágil

Una vez definido lo que es una metodología y sus variantes, la investigación se centra en las Metodologías Ágiles, en donde es importante tener claro el concepto de **agilidad**. El cuál a continuación, se presenta:

> *"La agilidad es un comportamiento persistente o habilidad, de entidad sensible, que presenta flexibilidad para adaptarse a los cambios esperados o inesperados, rápidamente; persigue la duración más corta en tiempo, usa instrumentos económicos; y utiliza los conocimientos y experiencias previos para aprender tanto del entorno interno como del externo." -Asif Qumer, Brian Henderson-Sellers, Tom McBride. [6]*

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

En otras palabras, la agilidad se enfoca en el valor de las personas al hacer eficazmente el trabajo y tener la habilidad de la **adaptabilidad** durante la creación de un producto, servicio u otro resultado cuando surjan cambios inesperados.

Tomando esta definición como punto de partida, se creó el Manifiesto Ágil:

"Estamos descubriendo formas mejores de desarrollar software tanto por nuestra propia experiencia como ayudando a terceros. A través de este trabajo hemos aprendido a valorar:

Individuos e interacciones** sobre **procesos y herramientas
Software funcionando** sobre **documentación extensiva
Colaboración con el cliente** sobre **negociación contractual
Respuesta ante el cambio** sobre **seguir un plan

Esto es, aunque valoramos los elementos de la derecha, valoramos más los de la izquierda." [8]

El manifiesto describe cuatro valores importantes:

1. **Individuos e interacciones sobre Procesos y herramientas.**
   Los individuos e interacciones son esenciales para un equipo de alto rendimiento, se ha comprobado mediante el estudio de "Saturación de la Comunicación" que mientras no exista ningún problema de comunicación los equipos pueden rendir hasta 50 veces más, el valor se centra en las personas más allá de los procesos y herramientas a utilizar dentro de la realización de un proyecto.

2. **Software funcionando sobre documentación extensiva.**
   El siguiente valor nos describe la importancia de la entrega del software, producto o servicio funcionando. La documentación pasa a un segundo plano, sin dejar de determinar la definición de **terminado**.

3. **Colaboración con el cliente sobre negociación contractual.**
   Las Metodologías Ágiles fomentan este valor, haciendo que un representante del cliente trabajé mano a mano con el equipo de desarrollo, dándole entregas del producto o servicio en periodos cortos de tiempo creando una ventaja de adaptabilidad al realizar mejoras sugeridas por el cliente, dejándolo plenamente satisfecho, a diferencia de una negociación contractual en donde el cliente únicamente se hace presente al inicio del proyecto esperando que el equipo pueda lograr la visión del proyecto tal y como lo deseaba.

4. **Respuesta ante el cambio sobre seguir un plan.**
   Este valor implica adaptar los planes a intervalos regulares en función de los comentarios del cliente, en las metodologías tradicionales los proyectos cuentan con una sola planificación que no contempla un cambio por parte del cliente, y para cuando finaliza el proyecto ya es demasiado tarde agregar los cambios, dejando insatisfecho y decepcionado al cliente, es por lo que las metodologías ágiles valoran más la respuesta al cambio para tener mayor éxito en sus proyectos. [6]

2.5 Principios del Manifiesto Ágil

El manifiesto ágil, tras los postulados de estos cuatro valores en los que se fundamenta, establece estos 12 principios [8]:

1. Nuestra principal prioridad es satisfacer al cliente a través de la entrega temprana y continua de software de valor.

Satisfacción del cliente

2. Son bienvenidos los requisitos cambiantes, incluso si llegan tarde al desarrollo. Los procesos ágiles aceptan el cambio y lo toman como ventaja competitiva para el cliente.

Bienvenidos los cambios en los requerimientos

3. Entregar con frecuencia software que funcione, en periodos de un par de semanas hasta un par de meses, con preferencia en los períodos breves.

Entrega de Producto frecuentemente

4. Las personas del negocio y los desarrolladores deben trabajar juntos de forma cotidiana a través del proyecto.

Colaboración Diaria

5. Construcción de proyectos en torno a individuos motivados, dándoles la oportunidad y el respaldo que necesitan y procurándoles confianza para que realicen la tarea.

Colaboradores Motivados

6. La forma más eficiente y efectiva de comunicar información de ida y vuelta dentro de un equipo de desarrollo es mediante la conversación cara a cara (*face to face).*

Comunicación Cara a Cara

7. El software que funciona es la principal medida del progreso.

Medición del Avance por Trabajo completado

8. Los procesos ágiles promueven el desarrollo sostenido. Los patrocinadores, desarrolladores y usuarios deben mantener un ritmo constante de forma indefinida.

Promover un ritmo sostenido

9. La atención continua a la excelencia técnica enaltece la agilidad.

Atención a la Excelencia

10. La simplicidad como arte de maximizar la cantidad de trabajo que se hace es esencial.

Simplicidad es Esencial

11. Las mejores arquitecturas, requisitos y diseños emergen de equipos que se auto organizan.

Equipo Auto-Organizados

12. En intervalos regulares, el equipo reflexiona sobre la forma de ser más efectivo y ajusta su conducta en consecuencia. [8]

Reflexionar en la mejora continua

Finalmente, la metodología ágil permite reforzar la comunicación entre los desarrolladores y el cliente, con el objetivo de que el producto satisfaga los requerimientos al momento de la entrega final.

Capítulo III. Scrum

3.1 Descripción de Scrum

3.1.1 Historia

El concepto de Scrum se conoció por primera vez en **1986**, por un estudio llamado "*The New Product Development Game*" publicado por **Takeuchi** y **Nonaka**, en donde se observaron los procesos innovadores de desarrollo en productos exitosos de compañías como Honda, HP, Canon, etcétera. Dichos procesos permitían el desarrollo de un producto en menos tiempo, de buena calidad y a un bajo costo.
La diferencia radicaba en que se contaba con un único equipo multidisciplinario que atendiera todas las fases de desarrollo del proyecto y no con un equipo diferente para cada fase como en otras industrias.
Esta técnica de tener un sólo equipo de desarrollo que se apoyará desde los inicios del proyecto hasta la entrega del producto final fue comparada con la famosa jugada llamada "***Scrum***" del juego de Rugby, en la cual la colaboración y comunicación que tienen los jugadores los lleva a la victoria. [6] A continuación, se muestra en la *Imagen B* lo anteriormente descrito:

Anónimo. (2019). Estrategia "Scrum" en Rugby. [Imagen B]. Recuperado de: https://pngio.com/images/png-a79925.html

En **1993**, **Jeff Sutherland** creó, ejecutó y documentó el primer modelo de Scrum para el Desarrollo Ágil de Software en Easel/Corporation, utilizando el estudio de gestión de equipos de Takeuchi y Nonaka como base. [6]

En **1996**, **Jeff Sutherland**, en colaboración con **Ken Schwaber,** presentaron formalmente las prácticas que se usarían como proceso final para la industria del Desarrollo de Software llamando al proceso **Metodología Scrum** que pasaría a incluirse en la lista de Metodologías Ágiles, como se observa en el capítulo II. [6]

3.1.2 Propósito

El principal propósito de *Scrum* es abordar problemas complejos de cualquier índole, mediante el desarrollo, entrega y mantenimiento de los productos finales. Los productos que se entregan son de máximo valor para el cliente. [4]

3.1.3 Aplicaciones de Scrum

Desde principios de los años 90 se ha utilizado ampliamente en todo el mundo para:

1. Software,
2. Hardware,
3. Escuelas,
4. Gobiernos,
5. Mercadeo,
6. Gestión de Operaciones, etcétera.

Está estructurado de tal manera que es compatible con los productos y el desarrollo de servicios en todo tipo de organizaciones y en cualquier tipo de proyecto, independientemente de su complejidad. [4]

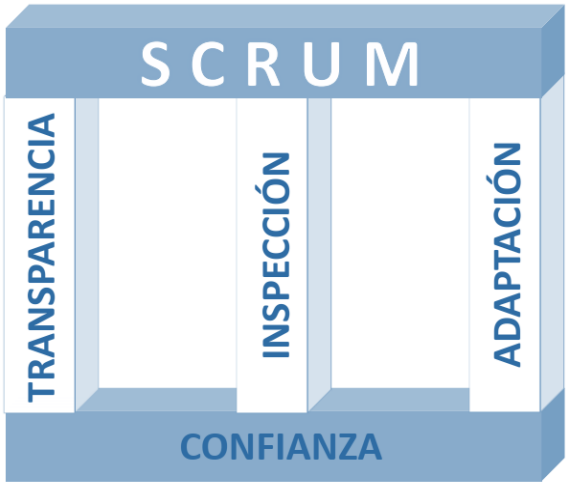
3.1.4 Teoría de Scrum

Scrum se basa en la **teoría de control de procesos empírica** [4], que consiste en la toma de decisiones basadas en la experiencia de los integrantes del equipo, abriendo un camino hacia un incremento (mejoramiento) iterativo de conocimiento y una optimización en la predictibilidad de riesgos durante el proyecto.

Para que la teoría de control de procesos empírica se implemente de una forma correcta, no hay que omitir los tres pilares fundamentales: **transparencia, inspección y adaptación.** [4]

A continuación, en la *Imagen C* se puede observar la importancia de estos pilares en la metodología *Scrum*:

Pilares de Scrum



ON-TIME. (2020). Pilares de Scrum. [Imagen C]. Recuperado de: https://on-time.es/portal/productividad-colectiva/agile/scrum/

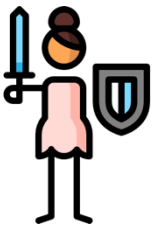
- **Transparencia**
  Todos los integrantes del equipo que estén involucrados en los procesos de desarrollo deben de tener un entendimiento común de los aspectos significativos a realizar.

- **Inspección**
  Se debe inspeccionar frecuentemente los entregables salientes durante los procesos del proyecto para detectar posibles errores o variaciones no deseadas en los resultados finales.

- **Adaptación**
  Si en una inspección se determina que uno o más aspectos del proceso no cumplen con los objetivos del proyecto, esto debe de ajustarse cuanto antes para evitar desviaciones mayores.

Estos tres pilares son muy importantes dentro de un equipo de Scrum, pero para fomentarlos es necesario hacer uso de **los valores de Scrum.** A continuación, se presentan dichos valores en la *Tabla 1*:

Tabla 1: Valores de Scrum

Valor	Descripción
Foco 	Enfocarse durante el desarrollo del proyecto para lograr los objetivos.
Coraje 	Asumir compromisos desafiantes.
Apertura 	Transparencia y sinceridad en nuestro trabajo diario con el equipo.
Compromiso 	Cada integrante debe comprometerse con sus tareas para lograr el éxito del equipo.
Respeto 	El respeto mutuo dentro del equipo.

3.1.5 Scrum vs. Gestión tradicional de proyectos

Ambas metodologías son importantes para el desarrollo de proyectos. Sin embargo, tienen enfoques diferentes dentro de la solución de un problema. En la *Tabla 2* se pueden apreciar las diferencias entre los principales enfoques durante el desarrollo de un proyecto [7]:

Tabla 2: Diferencias entre Scrum y la gestión tradicional de proyectos

Enfoque	Scrum	Gestión tradicional de proyectos
El énfasis está en	Personas	Procesos
Documentación	Sólo mínima (Según se requiera)	Exhaustiva
Estilo de Procesos	Iterativo	Lineal
Planificación por adelantado	Baja	Alta
Priorización de los requisitos	Según el valor del negocio y regularmente actualizada	Fijo en el plan de proyecto
Garantía de calidad	Centrada en el cliente	Centrada en el proceso
Organización	Auto organizada	Gestionada
Estilo de gestión	Descentralizado (No hay un líder)	Centralizado (Existe un líder)
Cambio	Actualizaciones a la lista priorizada de pendientes del producto	Sistema formal de gestión del cambio
Liderazgo	Liderazgo colaborativo y servicial (Scrum Master: Colabora con todos y ayuda, líder servicial)	Mando y Control (Project Management)
Medición del rendimiento	El valor del negocio	Conformidad con el plan
Retorno sobre la Inversión	Al comienzo y a lo largo del proyecto	Al final del proyecto
Participación del cliente	Alta durante todo el proyecto	Varía en función del ciclo de vida del proyecto.

3.2 Elementos de Scrum

Scrum es una metodología de adaptación iterativa, rápida, flexible y eficaz. Está diseñada para ofrecer un valor significativo de forma rápida al inicio del proyecto. Está estructurada principalmente de 3 elementos: **principios, procesos y aspectos.** A continuación, en la *Imagen D*, podemos observar la forma colaborativa de los elementos durante el desarrollo de la metodología [7]:

Imagen D: Elementos de Scrum

3.2.1 Principios

Los principios de *Scrum* se deben abordar y gestionar durante todo el proyecto, no se pueden modificar, ni mucho menos omitirlos. Se clasifican en: Control de proceso empírico, Auto-organización, Colaboración, Priorización basada en valor, Asignación de un bloque de tiempo (*Time-Boxing*) y Desarrollo iterativo. [7]

- **Control de proceso empírico**
  Scrum prescribe la toma de decisiones con base en la observación y experimentación en vez de planeaciones iniciales detalladas, enfocándose en la adaptabilidad de la metodología al proyecto.

- **Auto-organización**
  Este principio se centra en los trabajadores, y la entrega significativa de valor a su trabajo cuando se organizan a sí mismos, dando un sentido de compromiso y responsabilidad dentro del equipo, este ambiente es más propicio para el crecimiento.

- **Colaboración**
  El desarrollo del producto es un proceso compartido, que requiere del trabajo y colaboración conjunta de todos los miembros con la finalidad de entregar el mayor valor posible, cada integrante debe sentirse importante y que sus opiniones y decisiones son valiosas.

- **Priorización basada en valor**
  Este principio prioriza el máximo valor del negocio, trabajando desde el principio del proyecto hasta su conclusión con lo que le da mayor valor de negocio, generando un retorno sobre la inversión rápido.

- **Asignación de un bloque de tiempo (*Time-Boxing*)**
  Los *Time-Boxing* o Bloques proponiendo la fijación de una cierta cantidad de tiempo para cada proceso y actividad dentro del proyecto.

- **Desarrollo iterativo**
  Los cambios se deben manejar iterativamente para crear productos que satisfagan las necesidades cambiantes y constantes del cliente.

3.2.2 Aspectos

A diferencia de los principios, los aspectos no son esencialmente obligatorios, pero si deseables durante el proyecto. Se dividen en: Organización, Justificación del negocio, Calidad, Cambios y Riesgo. [7]

- **Organización**
  Es importante porque permite visualizar la estructura organizacional dentro del plan de negocio.

- **Justificación del negocio**
  El proyecto debe estar soportado por un conjunto de beneficios o valores que permita mejorar aspectos del negocio del cliente.

- **Calidad**
  Asegurar la calidad del producto o servicio a partir de incrementos parciales hasta llegar al producto o servicio final.

- **Cambios**
  Hacer frente al cambio sin afectar utilidades y beneficios del producto o servicio obtenido una vez finalizado el proyecto.

- **Riesgo**
  El riesgo según su impacto en el proyecto puede ser vistos como oportunidades cuando le genera valor al proyecto.

3.2.3 Procesos

Los procesos de Scrum abordan las actividades y el flujo de trabajo durante el proyecto. En total hay diecinueve procesos que se agrupan en cinco fases.
A continuación, se muestran en la *Tabla 3.* [7]

Tabla 3: Fases y Procesos de Scrum

Fases	Procesos
Inicio	1. Creación de la visión del proyecto. 2. Identificación del *Scrum Master* y el/los socio(s). 3. Formación de Equipos de *Scrum*. 4. Desarrollo de Épica(s). 5. Creación de la lista priorizada de pendientes del producto (*Product Backlog*) 6. Realizar la planificación de lanzamiento.
Planificación y Estimación	7. Creación de historias de usuario. 8. Estimación de historias de usuario. 9. Comprometer historias de usuario. 10. Identificación de tareas. 11. Estimar tareas. 12. Creación de la lista de pendientes del *Sprint* (*Sprint Backlog*).
Implementación	13. Creación de entregables. 14. Realizar reunión diaria de pie (*Daily Standup*). 15. Mantenimiento de la lista priorizada de pendientes del producto (*Backlog Refinement*).

Revisión y Retrospectiva	16. Demostrar y validar *Sprint* (Sprint Review). 17. Retrospectiva del *Sprint* (Sprint Retrospective).
Lanzamiento	18. Envío de entregables. 19. Retrospectiva el proyecto.

3.3 Roles de Scrum

Los roles de *Scrum* se dividen en dos grandes categorías, centrales y no centrales.

3.3.1 Roles centrales

Son las personas que están comprometidas con el proyecto y el proceso de Scrum.

Dueño del Producto *(Product Owner)*

- Responsable de lograr el máximo valor de negocio para el proyecto.
- Articula los requerimientos del cliente.
- Mantiene la justificación del negocio para el proyecto.
- Representa la voz del cliente. [7]

Scrum Master

- Asegurar que el equipo de Scrum cuente con el ambiente adecuado para completar con éxito el proyecto.
- Guía, facilita y enseña las prácticas de Scrum a todos los involucrados.
- Elimina obstáculos en el equipo.
- Se asegura de que se encuentren siguiendo los procesos de Scrum. [7]

Equipo Scrum *(Scrum Team)*

- Responsables de entender los requerimientos especificados por el *Product Owner*.
- Crear los entregables del proyecto.
- Estiman el esfuerzo de las tareas a realizar en el proyecto. [7]

3.3.2 Roles no centrales

Aunque no son parte del proceso de *Scrum*, es necesario que formen parte de la retroalimentación de los resultados del proyecto, para poder revisar y planear cada periodo de trabajo de manera colaborativa:

Usuarios

- Es el destinatario final del producto.

Clientes

- Puede ser uno o varios quienes actúan como los solicitantes del producto.

Patrocinadores (*Stakeholders*)

- Las personas a las que el proyecto les producirá un beneficio. Participan durante la revisión del *Sprint* (*Sprint Review*), y también pueden interactuar como usuarios finales o clientes al mismo tiempo. [7]

3.4 Desarrollo de las fases de un proyecto Scrum

Las fases de un proyecto de *Scrum* se dividen en: Fase de inicio, Fase de planificación y estimación, Fase de implementación, Fase de revisión y retrospectiva y Fase de lanzamiento.

3.4.1 Fase de inicio

A continuación, en la *Tabla 4* se presentan los procesos de la fase de inicio: Creación de la visión del proyecto, Identificación del *Scrum Master* y *Stakeholders*, Formación de equipos *Scrum*, Desarrollo de épicas, Creación del *Backlog* y Planificación de lanzamiento. [7]

Tabla 4: Fase de Inicio de Scrum

Proceso	Descripción
	Explica la necesidad empresarial que el proyecto busca cumplir y debe enfocarse en el problema en vez de la solución. Reunión: Visión del proyecto. Artefacto: Declaración de visión del proyecto.

	En esta fase se identifica a la persona que estará ejerciendo el cargo como *Scrum Master* y los socios del proyecto.
	Se integra al capital humano los roles centrales del proyecto.
	Una vez definidos los roles, es necesario que el *Product Owner* se encargue del desarrollo de Épicas. Las épicas son historias de usuario no desarrolladas o, en otras palabras, bloques con tareas más pequeñas.
	En este proceso se priorizan las épicas para crear una lista priorizada de pendientes del producto y se establecen los criterios de terminado. Este trabajo lo sigue realizando el experto en el negocio: *Product Owner.* Artefactos: Lista priorizada de pendientes del producto (*Product Backlog).*
	En esta parte se realiza una reunión de revisión en donde todos los integrantes principales del equipo de *Scrum* revisan las épicas para realizar un cronograma de planificación del lanzamiento en su versión 1.0 del producto. Reunión: Planificación de lanzamiento.

3.4.2 Fase de planificación y estimación

A continuación, en la *Tabla 5* se presentan los procesos dentro de la fase de planificación y estimación:
Creación de historias de usuario, Estimación de historias de usuario, Comprometer de historias de usuario, Identificación de tareas, Estimar tareas y Creación del *Sprint Backlog*. [7]

Tabla 5: Fase de planificación y estimación de Scrum

Proceso	Descripción
	En este proceso se crean las historias de usuario, que son épicas desarrolladas. Están diseñadas para asegurar que los requisitos del cliente estén claramente representados y finalmente son incorporadas al *Product Backlog*. Las historias de usuario deben de responder la siguiente oración: "Yo **como** [rol del usuario] **quiero** [Objetivo] **para poder** [beneficio]".
	Una vez terminadas las historias de usuario, el Scrum Master se encarga de realizar una reunión para la estimación de esfuerzo de las historias de usuario con el equipo de desarrollo. Aquí es donde se hace uso del Planning Poker, en donde cada tarjeta está basada en la serie Fibonacci. Cada tarjeta representa el grado de complejidad (esfuerzo) necesario para completar una tarea y cada integrante del equipo deberá mostrar la carta elegida para asignar a la tarea una estimación de complejidad. Finalmente, se asigna a la tarea el número que más aparece entre los integrantes. Reunión: Estimación de historias de usuario.
	En este proceso, el equipo de desarrollo se compromete y se hacen responsables de las historias de usuario ya estimadas. Reunión: Estimación de Historias de usuario.

	Las historias de usuario probadas, estimadas y asignadas se dividen en tareas específicas y se compilan en una lista de tareas. Reunión: Reunión de planificación y estimación de tareas. Artefacto: Lista de tareas actualizadas.
	Durante la reunión de planificación de tareas, el equipo de Scrum estima el esfuerzo necesario para realizar cada tarea en la lista. Aquí es donde también se hace uso del *Planning Poker.*
	En este proceso, el equipo principal de *Scrum* lleva a cabo una reunión de planificación del *Sprint.* El *Sprint* es un bloque de tiempo de un mes o menos durante el cual se crea un incremento del producto “Terminado” y se cumple una meta en particular (*Sprint Goal).* Reunión: Planificación del Sprint *(Sprint Planning)* Artefacto: Lista de Pendientes del Sprint *(Sprint Backlog*) y Diagrama de *Burndown Chart*. Este diagrama es una representación gráfica del trabajo realizado en un proyecto durante el *Sprint*. Este diagrama es útil para predecir si las tareas se terminaran a tiempo.

3.4.3 Fase de implementación

A continuación, en la *Tabla 6* se presentan los procesos dentro de la fase de implementación: Creación de entregables, Realización del *Daily Standup* y Refinación del *Backlog*. [7]

Tabla 6: Fase de implementación de Scrum

Proceso	Descripción
	El equipo de Scrum trabaja en el *Sprint Backlog* para crear los entregables, generalmente se utiliza un tablero de *Scrum (Scrum Board)* para dar seguimiento al trabajo y las actividades que se llevan a cabo y tener transparencia durante el proceso. Reunión: Planificación del Sprint (*Sprint Planning)* Artefacto: *Scrum Board*
	En este proceso, se lleva a cabo diariamente una reunión altamente focalizada con un bloque de tiempo asignado de 15 minutos llamada *Scrum* Diario *(Daily Standup).* Es un foro para que el equipo se ponga al día sobre sus progresos e obstáculos *(Blockers)* que puedan presentar y se requiera la ayuda del *Scrum Master*. Reunión: Scrum Diario *(Daily Standup)* Artefacto: S*crum Board* y Diagrama de *Burndown* actualizados, Registro de Obstáculos (*Sprint Blockers)*
	Constantemente se debe actualizar y dar mantenimiento al *Product Backlog* del proyecto.

3.4.4 Fase de revisión y retrospectiva

A continuación, en la *Tabla 7* se presentan los procesos dentro de la fase de revisión y retrospectiva: Demostración y validación del *Sprint* y Retrospectiva del *Sprint*. [7]

Tabla 7: Fase de revisión y retrospectiva de Scrum

Proceso	Descripción
	El equipo de Scrum demuestra los entregables del *Sprint* al *Product Owner* durante la Revisión de *Sprint (Sprint Review),* el propósito de esta reunión es lograr la aprobación y aceptación respecto al producto o servicio esperado. Reunión: Revisión de Sprint *(Sprint Review)*
	En esta parte, el *Scrum Master* y el equipo de desarrollo se reúnen para discutir las lecciones aprendidas durante el *Sprint*. Dicha información se documenta como mejoras accionables aceptadas llamadas *(Agreed Actionable Improvements)* que servirán para el siguiente *Sprint*, en el incremento iterativo durante el proyecto. *Reunión: Retrospectiva de Sprint (Sprint Retrospective)*

3.4.5 Fase de lanzamiento

A continuación, en la *Tabla 8* se presentan los procesos dentro de la fase de lanzamiento: Enviar entregables y Retrospectiva del proyecto. [7]

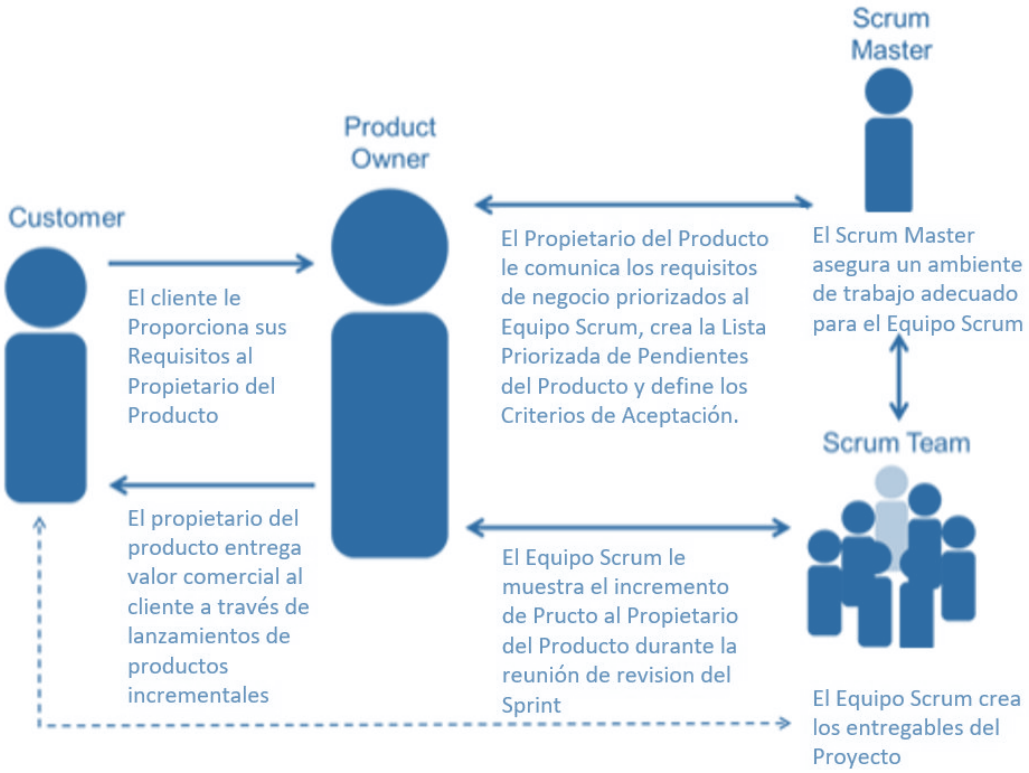
Tabla 8: Fase de lanzamiento de Scrum

Proceso	Descripción
	Se hace énfasis en la entrega al cliente de los entregables aceptados, la conclusión satisfactoria del sprint se documenta en un acuerdo formal de entregables funcionales.

	En este proceso final, se concluye el proyecto, los socios de la organización y los miembros del equipo principal de Scrum se reúnen para hacer una retrospectiva del proyecto e identificar, documentar las lecciones que se aprendieron, para mejora de futuros proyectos.

3.4.6 Esquema de comunicación Scrum

En la *Imagen E* se observa con más detalle los miembros del equipo Scrum y el papel que desempeñan durante el desarrollo del proyecto:



Anónimo. (2018). Esquema de comunicación Scrum. [Imagen E]. Recuperado de: http://gestiondeproyectos-master.com/roles-y-responsabilidades-en-un-proyecto-scrum/

3.4.7 Flujo de Scrum

A continuación, se muestra en la *Imagen F* el flujo que lleva la Metodología Scrum:

Romero, G. (2019). Flujo de trabajo Scrum. [Imagen F]. Recuperado de: https://platzi.com/clases/Scrum/

Como se observa, la **fase de inicio** desarrolla desde el caso de negocio del proyecto hasta el proceso de la priorización del backlog del producto, pasando por la realización de la visión del proyecto y el cronograma de lanzamiento.

La **fase de planificación y estimación** es la siguiente en hacerse presente en la *Imagen F*, se basa en la realización de la lista priorizada de tareas del *Sprint* o también llamada *Sprint Backlog* y la estimación por puntos de las tareas, si es el caso de ser necesario. Desde esta fase hasta la última, se realiza el proceso iterativo para encontrar mejoras en el desarrollo de los entregables.

La siguiente fase, llamada **fase de implementación**, es aquella en donde se lleva a cabo la realización del *Sprint*. Las tareas de la lista priorizada del sprint backlog son realizadas, dando como resultado entregables. Igualmente, se efectúan las ceremonias diarias (*Daily Standup*) donde se da un reporte breve a todo el Equipo *Scrum* y al *Scrum Master* sobre las tareas realizadas, las tareas por realizar y si existe algún impedimento o bloqueo para la realización de alguna tarea. Esta última parte es de suma importancia para mantener la **transparencia, inspección y adaptación** dentro del equipo.

Como penúltima fase se encuentra la **fase de revisión y retrospectiva** en la cual se proporcionan los entregables para su revisión a los interesados dentro del proyecto (*Poduct Owner y Stakeholders*), validando el trabajo realizado durante el *Sprint*; Igualmente, se da una retrospectiva del trabajo como equipo y del entregable para ver y mejorar las áreas de oportunidad en la siguiente Iteración-*Sprint*.

Finalmente, se encuentra la **fase de lanzamiento**. En donde se lleva a producción el producto resultante del trabajo realizado durante el proyecto. Cabe resaltar que el lanzamiento puede realizarse al finalizar la primer Iteración-*Sprint* como una prueba “demo” o al final de la última iteración-*Sprint*, todo dependerá de los interesados en el proyecto.

Capítulo IV. Solución Algorítmica de Problemas

4.1 Introducción a conceptos básicos

Los seres humanos realizan de manera cotidiana una serie de pasos, procedimientos o acciones que les permiten alcanzar un resultado para resolver un problema o llegar a una meta. En las Ciencias de la Computación no es muy diferente la manera en la que se resuelve un **problema.** Pero para el caso de un sistema formal, la expresividad de este se encuentra más acotada. En esta área, los problemas computacionales se resuelven mediante la programación. La **programación** es un conjunto de procedimientos que permite alcanzar la **solución** a un grupo de problemas, mediante el desarrollo de un **programa computacional** a través de un **lenguaje de programación**. La **SAP** permite plantear y resolver los problemas antes mencionados con ayuda de estrategias y técnicas para la implementación de un algoritmo. [9]
Para poder tener una más amplia de lo que es SAP es necesario definir los siguientes conceptos:

4.1.1 Problema

Comúnmente, se puede definir a un problema como un conflicto que se presenta. Un inconveniente para alcanzar objetivos en distintos ámbitos, desarrollando la necesidad de encontrar una solución, siguiendo un procedimiento, como lo son los algoritmos. En términos computacionales, el problema es el objeto de estudio, el cual tiene elementos y propiedades que forman parte de un dominio de datos. Es importante lograr la comprensión del problema para que, una vez que éste se haya entendido completamente, se puedan determinar de manera adecuada los recursos del dispositivo computacional para el algoritmo destinado y llegar a una solución. En el mundo computacional los tipos de problemas más importantes son de: clasificación, búsqueda, procesamiento de cadenas, problemas gráficos, problemas combinatorios, problemas geométricos y problemas numéricos, entre otros. [16]

4.1.2 Solución

Una solución es una acción o respuesta que le da resultado cuantitativo o cualitativo al problema que se había presentado. Una vez encontrada la solución, se podrá llegar a los objetivos. En las ciencias de la computación, la cual es el dominio de enfoque de esta investigación, las soluciones a los problemas de esta índole son instrucciones específicas para obtener respuestas. Es este énfasis en los procedimientos constructivos definidos con precisión lo que diferencia a la rama de

otras disciplinas. Considerando a los algoritmos como soluciones procesales a los problemas. [16]

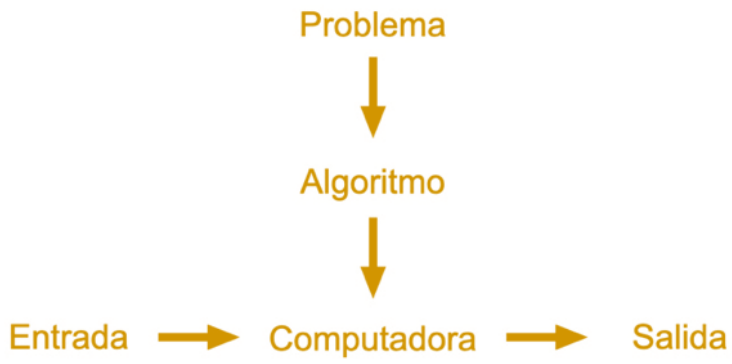
4.1.3 Algoritmo

Un tema central en las ciencias de la computación es el diseño de un proceso para llevar a cabo una tarea o varias tareas (subtareas) para resolver un problema; a este proceso se le conoce como algoritmo. Un algoritmo es aquel que se define como el conjunto de instrucciones que, al ser ejecutadas de manera ordenada, dan como resultado la solución de un problema. Por ejemplo, la tarea podría ser ordenar los nombres en orden alfabético, encontrar la forma más reducida de vincular un conjunto de sitios informáticos en una red, convertir un número a su representación en el sistema binario, cifrar un mensaje para una transmisión segura, diseñar un circuito digital para un microchip o determinar el camino más corto que debe seguir un brazo robótico. Se debe dejar clara la importancia, ya que diseñar un algoritmo es uno de los pasos para escribir un programa computacional. [16] También, cabe mencionar que pueden existir diferentes tipos de algoritmos para poder resolver una misma tarea, pero siempre será preferible que se optimice la salida en términos de complejidad computacional, es decir, espacial, número de cálculos y temporal (los tiempos y los recursos). Ejemplos de algoritmos esenciales en el mundo computacional son: *Hubble Sort, Shell Sort, Odd Even Sort, Insertion Sort y Selection Sort,* entre otros. [9]

Características:

- Preciso: Que su definición no dé lugar a ambigüedades sobre las instrucciones y conjunto de instrucciones para llegar a la solución de un problema.
- Determinista: No importando el número de veces que se repita, siempre llegará al mismo resultado.
- Finito: Debe tener fin en algún momento.
- Puede tener 1 o más elementos de entrada
- Debe producir un resultado: después de efectuar las instrucciones dará resultados de salida, aunque este pueda llegar a ser nulo. [9]

A continuación, en la *Imagen G* se presenta el proceso de planteamiento y resolución de un problema:



Levitin, A. (2012). Proceso de planteamiento y resolución de un problema. [Imagen G]. Introduction to the Design and Analysis of Algorithms. Villanova University: Pearson.

Clasificación:

Los algoritmos se clasifican en cualitativos y cuantitativos:

- **Cualitativos:** Se refiere un conjunto de algoritmos que predominantemente contienen instrucciones simbólicas, es decir, contiene pocos o casi ningún cálculo numérico. La mayoría de las aplicaciones de los algoritmos cualitativos quedan representados en la rama de la inteligencia artificial. Por ejemplo, en los sistemas de información geográfica se utiliza el razonamiento espacial cualitativo utilizando relaciones espaciales de distancia (norte, sur, este y oeste) para la georreferenciación. Los lenguajes de programación que usan este tipo de razonamiento son Prolog, Haskell, Lisp, Scheme, Python y, por supuesto, **Swift**.

A continuación, en el *Ejemplo I* se puede ver el desarrollo de un problema en donde una persona busca comprar unos zapatos de fiesta. Esta propuesta da un número finito y ordenado de pasos a seguir que se deben realizar para resolver el problema, cabe destacar que siempre se puede realizar de diferente manera y mejorar.

Algoritmo de la compra de zapatos de fiesta

Ejemplo I: Pasos a seguir del algoritmo de compra de zapatos de fiesta

1. INICIO
2. Entrar a la tienda y buscar la sección de zapatos de caballero.
3. Tomar un par de zapatos.
4. ¿Son zapatos de fiesta?
   SI: (ir al paso 5) – NO: (volver al paso 3)
5. ¿Hay de la talla adecuada?
   SI: (ir al paso 6) – NO: (volver al paso 3)
6. ¿El precio es pagable?
   SI: (ir al paso 7) – NO: (volver al paso 3)
7. Comprar el par de zapatos elegido.
8. FIN

- **Cuantitativos:** En este conjunto de algoritmos se busca la resolución del problema basándose predominantemente en cálculos numéricos. Este tipo de algoritmos son vistos en aplicaciones financieras, por ejemplo, operaciones bancarias, donde la solución es el cálculo del estado de cuenta de un usuario n. Lenguajes de programación utilizados para este fin son Cobol, C, C++, Java y, por supuesto, **Swift**.

A continuación, en el *Ejemplo II* se presenta el desarrollo de un problema en donde una persona busca obtener el área de un triángulo. Esta propuesta da un número finito y ordenado de pasos a seguir que se deben realizar para resolver el problema, cabe destacar que siempre se puede realizar de diferente manera y mejorar.

Algoritmo del área de un triángulo

Ejemplo II: Pasos a seguir del algoritmo del área de un triángulo

1. INICIO
2. Hallar las medidas de la base (b) y altura (h)
3. Multiplicar: base por altura (b x h)
4. Dividir entre 2 el resultado (b x h) / 2
5. FIN

Ahora, después de definir los conceptos, éstos se pueden unir y formar la siguiente ecuación como se muestran en la Imagen H:

Imagen H: Ecuación para encontrar una solución computacional

Es necesario tener un problema más un algoritmo podremos llegar a un resultado o solución de dicho problema. Esta ecuación representa el proceso de la **SAP.** [3]

4.2 Tipos de razonamiento

El razonamiento toma el conocimiento que ya se posee para resolver los problemas y la búsqueda de los algoritmos óptimos, pero ¿Cómo procesan exactamente las personas la información perceptiva del mundo que les rodea? Hay dos enfoques básicos para comprender cómo se produce esta sensación y percepción, Razonamiento Inductivo/*Top-Down y* Razonamiento Deductivo/*Bottom Up*:

- **Razonamiento Inductivo / *Top-Down***: Es aquel proceso del pensamiento en donde se va de lo particular a lo general, teniendo conclusiones globalmente ciertas, basadas en premisas muy particulares. Las problemáticas se resuelven por medio de alternativas probables que aportan una solución, es decir, los planteamientos no están predominantemente basados en la lógica, sino por el contrario, se fundamentan elementos estadísticos de posibles resultados favorables, frente a resultados negativos que pueden plantearse. [9] En mundo computacional, se entiende que la codificación no puede comenzar hasta que no se haya alcanzado un nivel de detalle suficiente, al menos en alguna parte del sistema.

- **Razonamiento Deductivo / *Bottom Up*:** Se basa en que se tiene una premisa considerada válida, y a partir de esta se obtiene una conclusión, va de lo general a lo particular. En los problemas los alumnos logran obtener la respuesta por medio de una serie de análisis y conjeturas fundadas en la razón y en el estudio de las circunstancias desde una perspectiva

primordialmente lógica y objetiva. [9] En el mundo computacional hace énfasis en la programación y pruebas tempranas, que pueden comenzar tan pronto se ha especificado el primer módulo. Este enfoque tiene el riesgo de programar cosas sin saber cómo se van a conectar al resto del sistema, y esta conexión puede no ser tan fácil como se creyó al comienzo. La reutilización del código es uno de los mayores beneficios del enfoque *bottom-up*.

4.3 Herramientas cognitivas

Para lograr resolver un problema, se necesita la creación de un algoritmo, para lograr esto se requiere de las herramientas cognitivas, pero primero hay que definir lo que son; Las **herramientas cognitivas** son aquellas que permiten describir de manera esquematizada, que apoyan, guían y extienden los procesos de pensamiento de los alumnos, para la construcción del conocimiento. Pueden ser utilizadas en una variedad de dominios específicos. Igualmente les dan a los alumnos la capacidad de un pensamiento crítico, creativo y colaborativo. Esta construcción del conocimiento es personal, ya que se construye sobre conocimientos previos, experiencias y creencias, se construye sobre la manera de interpretar el problema. En la solución algorítmica de problemas existen un gran número de representaciones. El enfoque se dará en dos de estas herramientas cognitivas: **Diagramas de Flujo** y **Pseudocódigo**. [9]

4.3.1 Diagrama de flujos

Son la representación gráfica de un algoritmo, mostrando gráficamente los pasos a seguir para lograr una solución y pueden contener varios caminos. Para la construcción de diagramas de flujo se deben tener en cuenta ciertas normas y figuras que se presentan a continuación:

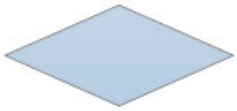
Reglas

1. Todo diagrama de flujo debe tener un principio y un fin.
2. Las líneas utilizadas para indicar el flujo del diagrama deben ser rectas, verticales y horizontales. (No deben cruzarse)
3. Todas las líneas utilizadas para indicar el flujo del diagrama deberán estar conectadas a un símbolo.
4. El diagrama tiene una construcción de arriba hacia abajo y de izquierda a derecha.
5. La notación utilizada en el diagrama de flujo es independiente a cualquier lenguaje de programación. [9]

Simbología

A continuación, en la *Tabla 9* se describen los símbolos utilizados en los diagramas de flujo. [13]

Tabla 9: Símbolos en diagramas de flujo

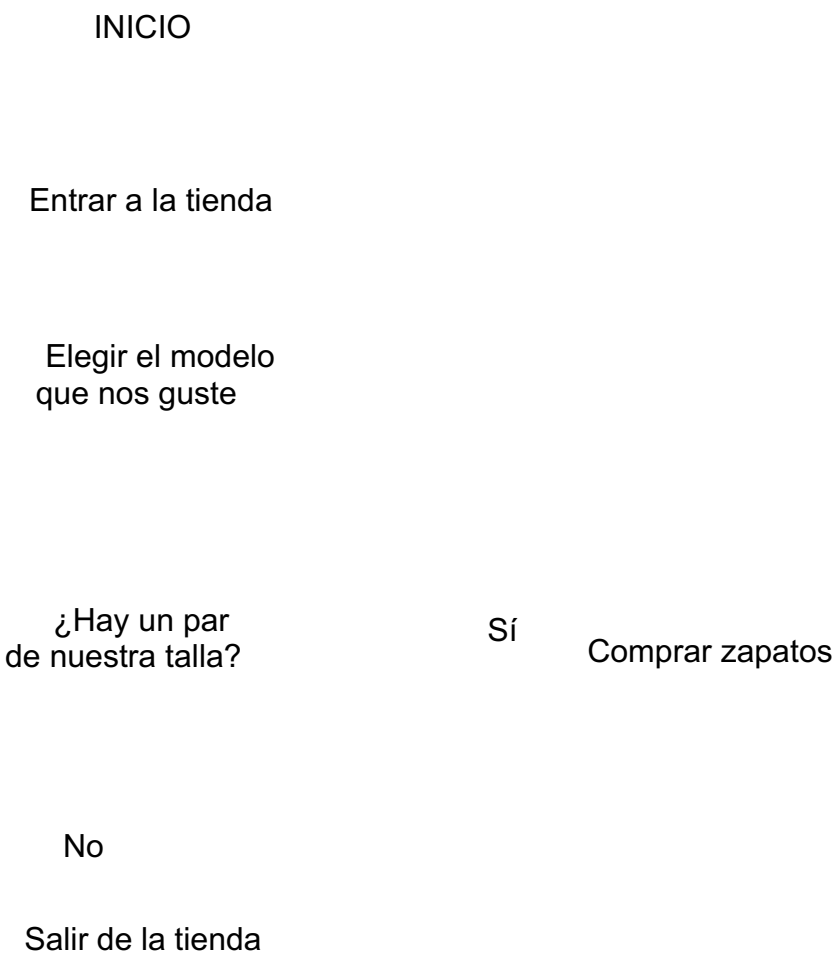
Símbolo	Nombre	Función
	Inicio / Final	El símbolo de terminación marca el punto inicial o final del sistema. Por lo general, contiene la palabra "Inicio" o "Fin". También puede que “Fin” sea representado como un círculo.
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa el material o la información que entra o sale del sistema, como una orden del cliente (entrada) o un producto (salida).
	Entrada Manual	Representa un paso donde se pide al usuario que introduzca la información manualmente.
	Proceso	Símbolo para representar un proceso. En su interior debe ir una asignación, operación, cambio de valor, etcétera.
	Decisión	Símbolo utilizado para representar una decisión. En su interior debe ir una condición la cual debe ser evaluada, dependiendo del resultado se toma el camino verdadero o falso. Este símbolo se utiliza en las estructuras selectivas si y si entonces/sino.

	Impresión	Símbolo que expresa escritura o impresión y se utiliza para imprimir un resultado.

Estos símbolos apoyan la construcción del pensamiento lógico en los alumnos al momento de la realización de un algoritmo para llegar a la solución de un problema ya existente.

Tomando el *Ejemplo I* de algoritmos cualitativos y *Ejemplo II* de algoritmos cuantitativos de la sección 4.1.3, se puede representar los diagramas de flujo respectivamente de la siguiente manera:

Ejemplo III: Diagrama de flujo del algoritmo de elección de zapatos de fiesta



Ejemplo IV: Diagrama de flujo del algoritmo del área de un triángulo

INICIO

Medidas de la base (b)
y altura (h)

Multiplicar: base por
altura (b x h)

Dividir entre 2 el
resultado (b x h) / 2

Resultado

FIN

4.3.2 Pseudocódigo

El ***pseudocódigo*** es una forma de expresar los distintos pasos que va a realizar un ***programa***, de la forma más parecida a un ***lenguaje de programación***. Su principal función es la de representar por pasos la solución a un problema o algoritmo, de la forma más detallada posible, dando paso a la implementación en un lenguaje de programación. [3] El pseudocódigo no puede ejecutarse en una computadora, ya que entonces dejaría de ser pseudocódigo, como su propio nombre lo indica, la

palabra *pseudo* proviene del griego que significa *falso* [14], se trata de un *código falso*, escrito para que lo entienda el ser humano y no una máquina.

Tomando el *Ejemplo II* de algoritmos cuantitativos de la sección 4.1.3, se tiene el siguiente pseudocódigo:

Ejemplo V: Pseudocódigo para el algoritmo del área de un triángulo

```
Proceso Área
        Definir base (b), altura (h), área(a);
        área<-b * h;
        área<-área / 2;
        Escribir "El área del triángulo es:", área;
Fin Proceso Área
```

4.4 Etapas para la solución de un problema

Una vez ya definidos los conceptos básicos de la **SAP**, se comenzará por ver que, para resolver un problema y diseñar un algoritmo y por consecuencia llegar a una solución, hay que tener en cuenta las etapas de la resolución de un problema computacional. Esta visión general del problema permitirá encontrar un algoritmo que se adecue a las necesidades del dominio del problema. Las etapas de resolución de un problema se dividen en 6:

1. **Análisis del problema, definición y delimitación:** Considerar los datos de entrada, el proceso que debe realizar la computadora y los datos de salida.

2. **Diseño y desarrollo del algoritmo:** se utiliza pseudocódigo, diagramas de flujo, etcétera.

3. **Pruebas de escritorio:** Seguimiento manual de los pasos descritos en el algoritmo. Se hace con valores bajos (si es un algoritmo cuantitativo) y tiene como fin detectar errores.

4. **Codificación:** Selección de un lenguaje de programación y digitación del algoritmo, haciendo uso de la sintaxis y estructura gramatical del lenguaje seleccionado.

5. **Compilación y Ejecución:** El programa es compilado y ejecutado (si no tiene errores) por la máquina para llegar a los resultados esperados.

6. **Evaluación de resultados:** Obtenidos los resultados se los evalúa para verificar si son correctos. (Un programa puede arrojar resultados incorrectos aun cuando su ejecución no muestra errores). [3]

Las etapas de resolución de un problema se pueden visualizar mejor con la *Imagen J* que a continuación se muestra:

Imagen I: Etapas de la solución de un problema computacional

Capítulo V. Tecnologías Apple en la educación

Como se mencionó anteriormente, el objetivo de los ***iOS Development Lab*** es combinar el aprendizaje de las *tecnologías de cómputo de Apple* para incentivar a toda la comunidad universitaria a crear aplicaciones móviles que tengan fines educativos, sociales y hasta comerciales para el mundo. Una vez que las aplicaciones hayan sido terminadas, éstas podrán ser publicadas en la App Store de Apple. De esta manera, se tendrá como resultado alumnos emprendedores y competitivos dentro de las ramas científicas, tecnológicas y sociales; con habilidades en trabajo en equipo, comunicación, creación del pensamiento crítico y una auténtica conexión con las problemáticas del mundo, utilizando las matemáticas aplicadas. Pero para llegar a estos resultados se necesita un poco más de dichas tecnologías de cómputo de Apple. Se comenzará por describir qué es **Xcode**, **Playgrounds**, **lenguaje de programación Swift** y la relación entre ellos.

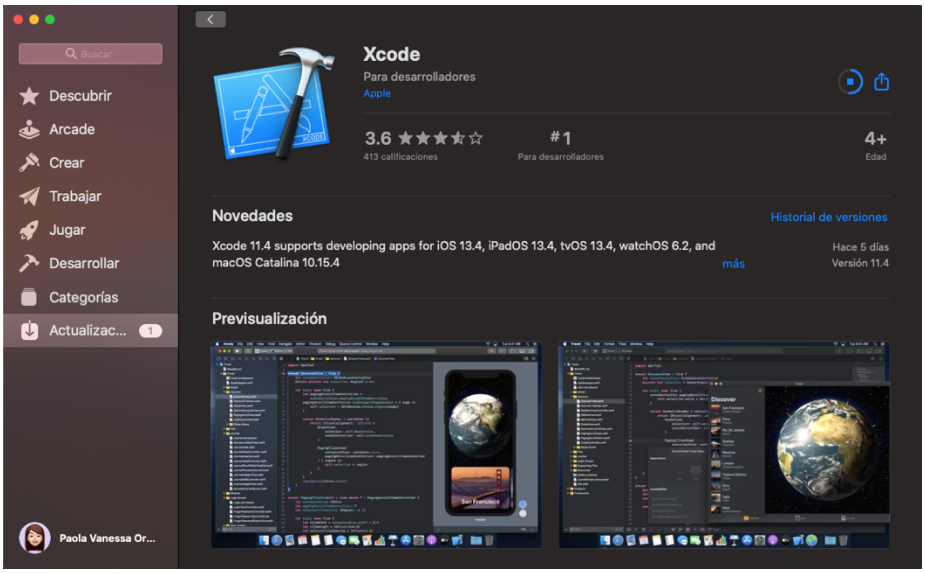
5.1 Xcode

Apple describe a Xcode como un entorno de desarrollo integrado (IDE) que contiene el conjunto de herramientas tecnológicas para el desarrollo de software para Mac, iPhone, iPad, Apple Watch y Apple TV. Esta tecnología es gratuita para su descarga desde la App Store. Sin embargo, se requiere de una cuenta de desarrollador. La versión más actual del entorno es 11.4. Con esta versión de Xcode se puede desarrollar desde los siguientes Sistemas Operativos: iOS 8 en adelante, tvOS 9 en adelante, watchOS 2 en adelante y con macOS Mojave 10.14.4 en adelante. [23]

En la *Imagen J se muestra el logo de Xcode:*

Apple Inc. (2020). Logo Xcode. [Imagen J]

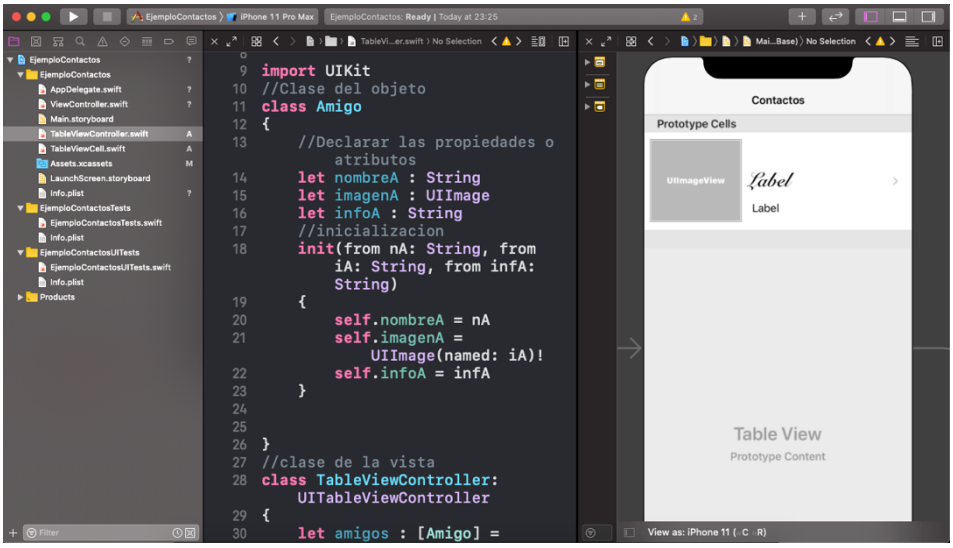
A continuación, en la *Imagen K* se muestra la Aplicación Xcode desde la App Store donde se observan sus características de desarrollo:



App Store. (2020). Aplicación Xcode. [Imagen K]

Dicha herramienta, creada para los programadores, combina las funcionalidades de diseño de la interfaz de usuario, programación, prueba y depuración de código en un flujo de trabajo unificado como se muestra en la *Imagen L*. También incluye una colección de **compiladores del proyecto GNU** (GCC) y entre los lenguajes de programación más importantes que puede compilar están: **C, C++, Objective-C y Swift.** [18]

Imagen L: Funcionalidades de Xcode



5.2 Playgrounds

Playgrounds es una herramienta de **prototipado de código** para inicios en el aprendizaje de la programación básica y orientada a objetos con el apoyo del lenguaje de programación **Swift.** Playgrounds también permite probar algoritmos computacionales sin necesidad de compilar una aplicación móvil completa, un *script*, etcétera.

A continuación, en la *Imagen M* se observa la pantalla de inicio de Xcode con la versión instalada, las 3 opciones de herramientas con las que se puede iniciar un nuevo proyecto y, marcada, la opción para crear un nuevo proyecto Playgrounds.

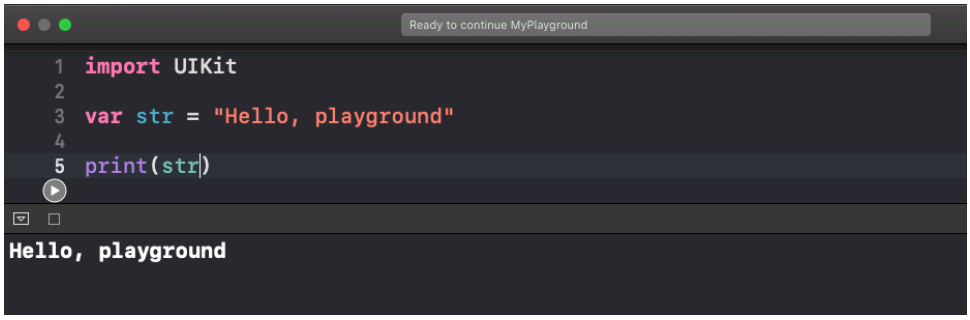
Imagen M: Inicio de un proyecto de Xcode

Como la mayoría de las tecnologías actuales, Swift puede ser programado en Playgrounds utilizando la filosofía REPL. Es decir, se puede realizar la conocida programación iterativa o también conocida como programación en tiempo real.
Entre las tecnologías análogas a Playgrounds se encuentra Jupyter[1], enfocada a los lenguajes de programación Python; Lisp, entre otros. [19]

[1] Project Jupyter. (2020). Project Jupyter. Sitio web: https://jupyter.org

En la *Imagen N* se muestra el modo donde se puede escribir y editar código a la vez que se obtiene un resultado en tiempo real:

Imagen N: Playground "Hola Mundo"



5.2.1 Elementos dentro de un Playground

Dentro de un Playground se encuentran diversos elementos para interactuar entre el código y el programador, los de mayor importancia son los siguientes [19]:

- **Sección de Codificación:** Aquí se encuentra el código editable, es decir, se puede escribir o modificar para lograr los resultados que se desee.
  En la *Imagen O* se señala este elemento:

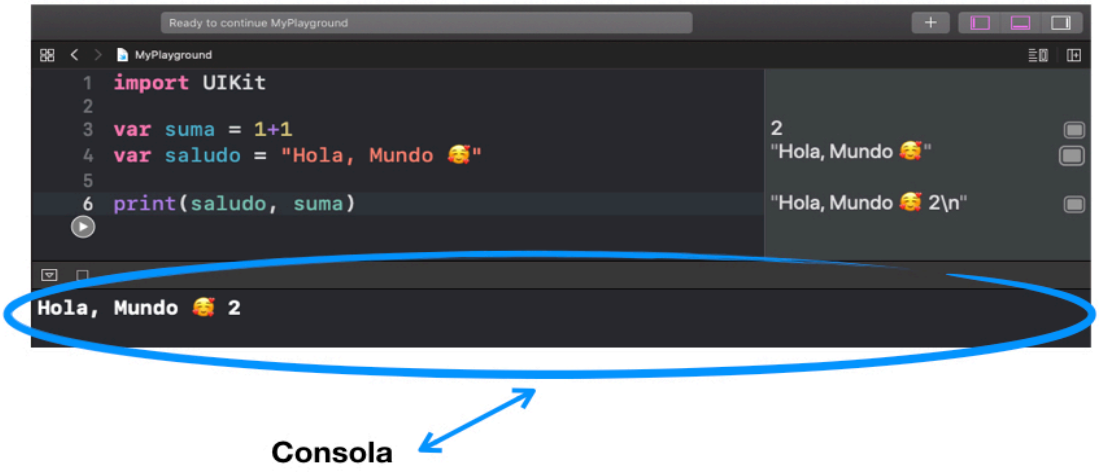
Imagen O: Sección de codificación en Xcode

- **Barra lateral de resultados:** A medida que se agrega código o se cambia, el Playground ejecuta el código constantemente y actualiza los datos en la barra lateral, no del resultado final de todas las líneas de instrucciones escritas en la sección de codificación, sino por cada línea de código. La barra lateral de resultados es un apoyo al momento de codificar las instrucciones para llegar a un resultado final exitosamente. En la *Imagen P* se señala este elemento:

Imagen P: Barra lateral de resultados en Xcode

- **Consola:** Es un centro de mensajes que muestra detalles sobre la forma en que se está ejecutando un programa. En la consola se puede imprimir con la instrucción *print* cualquier resultado que se haya querido mostrar con las instrucciones del código editable, confirmando los resultados ya mostrados en la barra lateral de resultados. En la *Imagen Q* se señala este elemento:

Imagen Q: Consola en Xcode



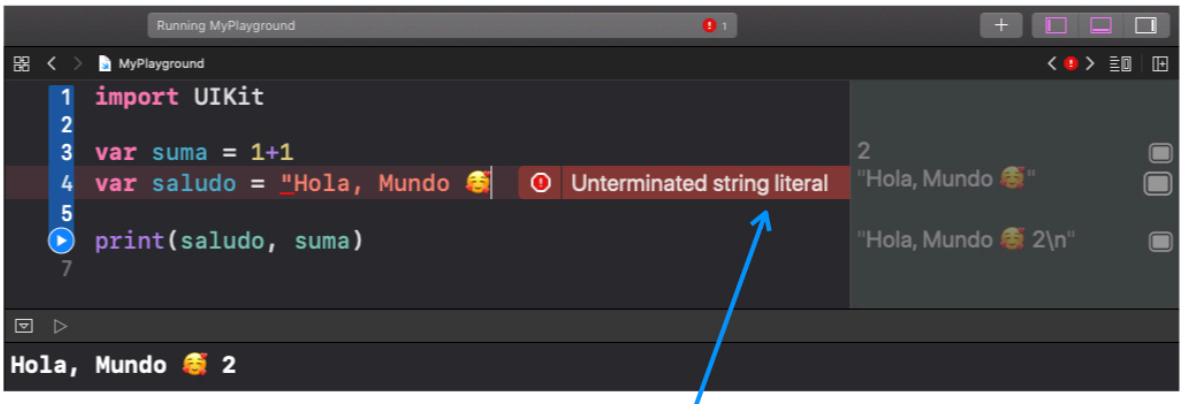
- **Comentarios:** Las notas dentro del código se denominan comentarios. El Playground ignora los comentarios, y éstos no afectan la ejecución del código. Un comentario comienza con dos barras // para una línea de código y con /**/ para varias, como se muestra en la *Imagen R*:

Imagen R: Ejemplo de comentario en Xcode

Comentario

- **Errores:** Cuando el Playground encuentra un error en el código, se resalta la línea de código en rojo en donde se encuentra dicho error y justo a la derecha aparece una descripción del error. El código detiene su ejecución. Dado que hay un error, no se muestran resultados en la barra lateral de resultados. En la *Imagen S* se señala este elemento:

Imagen S: Señal de error y descripción en Xcode



Señal de Error y Descripción

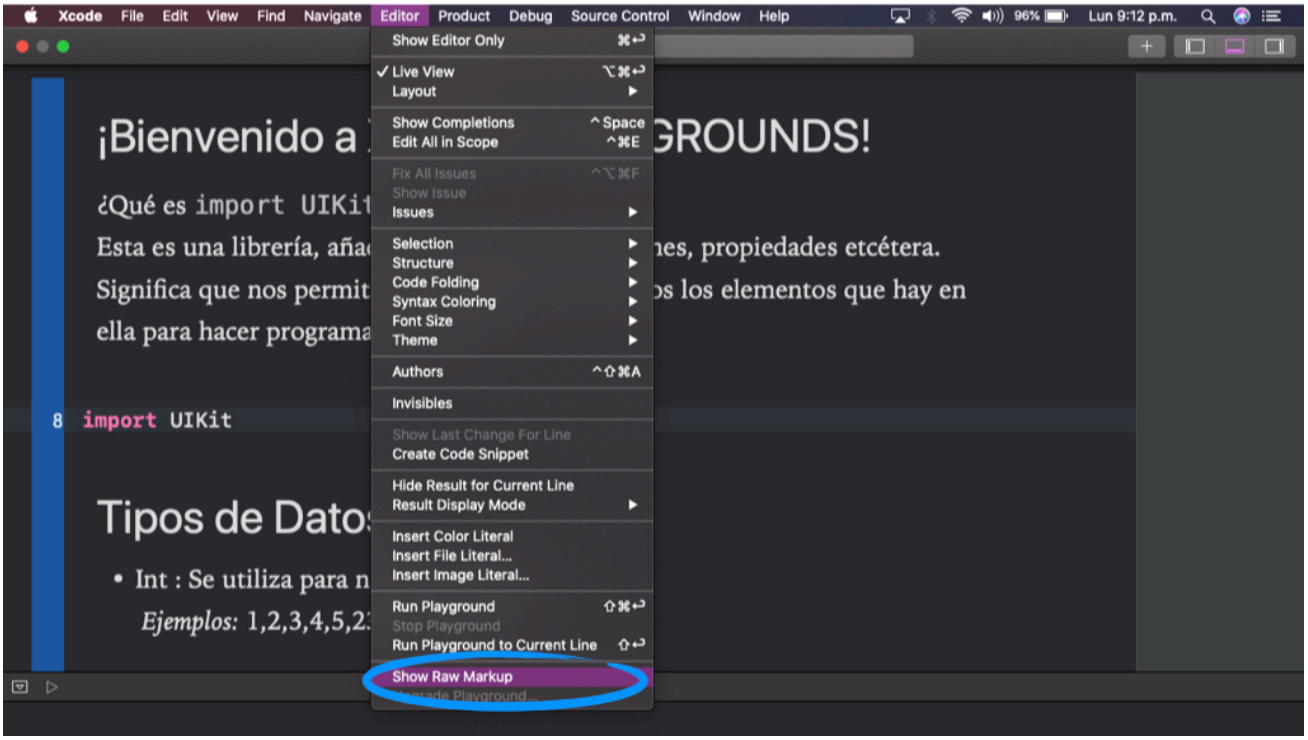
El objetivo de esta herramienta es que los estudiantes puedan experimentar con ideas y más adelante se puedan convertir en aplicaciones completas de funcionalidades mientras juegan y aprenden, esto lo hace ideal para la enseñanza como una herramienta tecnológica moderna.

5.2.2 Markup Language, lenguaje utilizado para la construcción de Playgrounds

Markup Language, o también llamado **lenguaje de marcado**, es un lenguaje formal que permite describir un documento Playground. Esto significa que permite combinar código de programación y elementos de apoyo multimedia como texto plano, imágenes, videos, audio y animaciones. [20]

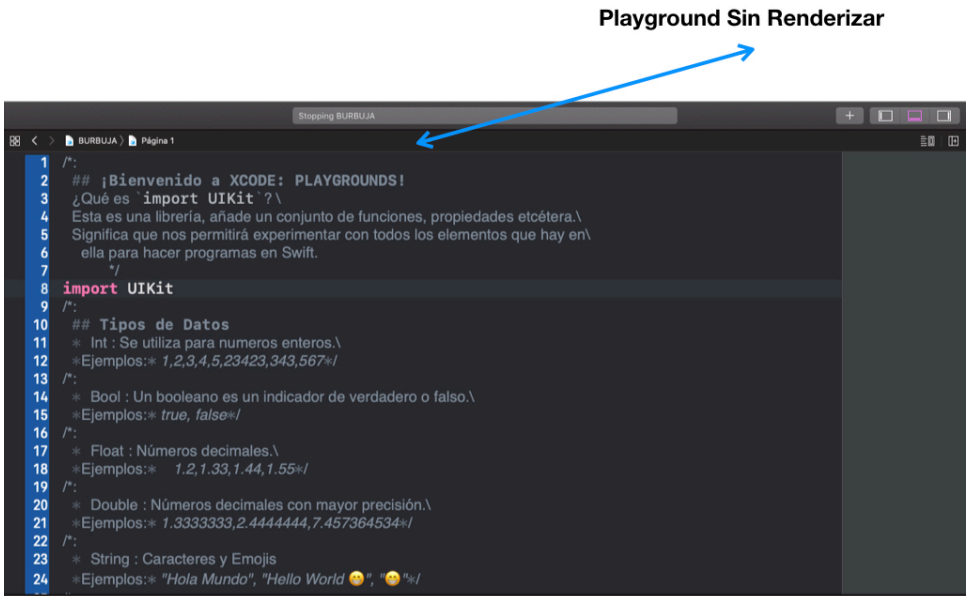
Para poder escribir instrucciones con el lenguaje de marcado en un documento Playground es necesario activar la siguiente opción. En la pestaña de ***Editor*** seleccionar ***Show Raw Markup*** como se muestra en la *Imagen T* marcada en color azul:

Imagen T Opción Markup Language en Playgrounds.



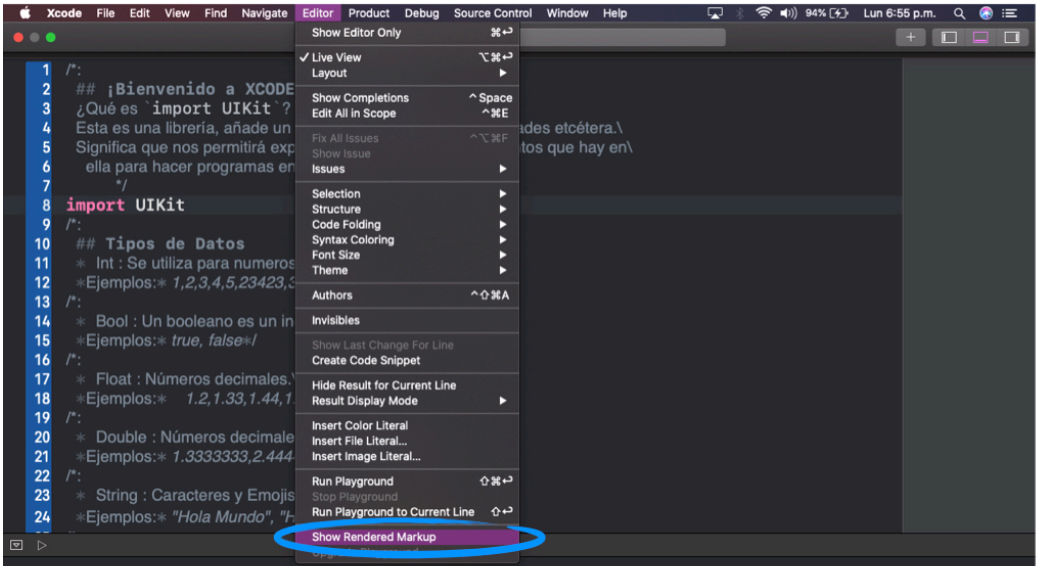
A continuación, en la *Imagen U* se muestra el lenguaje de marcado sin el procesamiento, es decir, sin la renderización del documento Playground:

Imagen U: Markup's en guía de aprendizaje: "Mis primeros pasos en Swift".



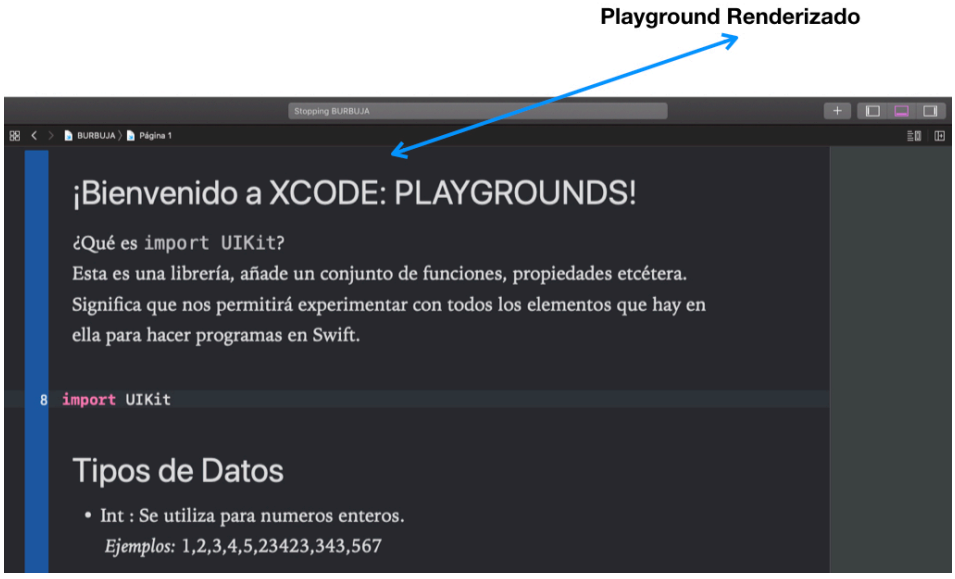
Posteriormente, para poder ver el documento Playground ya renderizado y con las ediciones del lenguaje de marcado, hay que seleccionar de nuevo la pestaña de **Editor**, seguido de la opción de **Show Rendered Markup**, como se muestra en la *Imagen V*:

Imagen V: Opción Markup Language en Playgrounds.



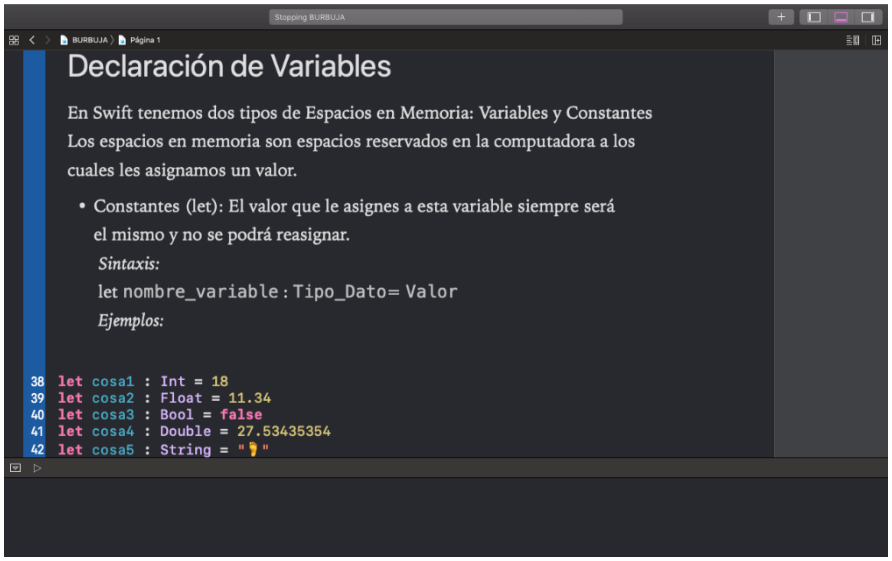
Mientras que, en la siguiente *Imagen W*, se muestra el Playground en modo de documentación renderizada o sin el lenguaje de anotación, para lo que compila el lenguaje y muestra los resultados de la caracterización del documento. Para poder utilizar este lenguaje de marcado en Playgrounds tenemos que hacer uso de la herramienta *Render Markup* y así poder comenzar con la caracterización del documento.

Imagen W: Playground renderizado



A nivel de herramienta de enseñanza didáctica, es posible mezclar documentación en modo lectura para una mejor comprensión del código con los diversos lenguajes que permite Xcode. A partir de ahora, el enfoque será únicamente hacia el lenguaje Swift. En la *Imagen X* se muestra lo anterior:

Imagen X: Lenguaje Swift en Playgrounds



5.3 Swift

Swift es un lenguaje de programación desarrollado en el año de 2014, con una versión actual 5.2. Un lenguaje altamente **tipado** (no permite el manejo entre variables de diferente tipo de dato), poderoso e intuitivo, creado con un enfoque moderno para mejorar la seguridad y rendimiento, basado en los patrones de diseño, fuertemente multiparadigma lo que significa que es orientado a protocolos, a objetos y a programación imperativa. Su logo se muestra en la *Imagen Y.* El objetivo del proyecto Swift es crear el mejor lenguaje disponible para la siguiente generación de desarrolladores. Algunos de sus usos se encuentran en la programación de sistemas de producción críticos, por ejemplo, los softwares bancarios, aplicaciones científicas, etcétera; Incluso aplicaciones móviles y de escritorio, ampliando su uso para servicios en la nube. [21]
Swift es de código abierto, disponible para desarrolladores, educadores y estudiantes. Debido a su fácil uso, es ideal para cualquier persona que desea aprenderlo en poco tiempo, además, para ayudar a que Swift sea un lenguaje aún más potente, existe una comunidad donde los usuarios pueden contribuir directamente a su código fuente llamada Swift.org. [22]

Apple Inc. (2020). Logo Swift. [Imagen Y]

Los desarrolladores no son los únicos que han notado el potencial de Swift. Universidades e instituciones académicas lo incluyen en sus cursos de programación y también ofrecen cursos gratuitos en iTunes U, KHan Academy, Platzi, Udemy, entre otras famosas. Esto significa que la transición de la programación básica a la profesional es más fácil. [21]

5.4 Unificación de la tecnología, herramienta y lenguaje Apple en la educación.

Como se ha observado, **Xcode** es una tecnología que contiene diversas herramientas para el desarrollo de programas en las distintas plataformas de Apple, para el sector educativo y principiantes en el lenguaje, o simplemente para pruebas de algoritmos. Para dicho fin, se puede utilizar la herramienta **Playgrounds**, en donde se puede combinar el uso del **Markup Language** para hacer más entendible el uso del lenguaje de programación **Swift** y lograr una enseñanza didáctica del mismo con complementos multimedia.

5.4.1 Guía de aprendizaje

Una vez revisados los conceptos anteriores, el presente trabajo ayuda a los estudiantes y profesores durante el proceso del aprendizaje a la programación dentro del ***iOS Development Lab*** de la F.E.S. ACATLÁN. La propuesta consistió en una guía de aprendizaje: ***"*Mis primeros pasos en Swift*"***, elaborada en la herramienta de Playgrounds, para el nivel básico del club de programación, en donde se incluyeran archivos multimedia, un temario base acompañado de ejemplos y actividades para los alumnos, tomando como referencia toda la investigación previamente realizada y basándose en la premisa de que un proyecto educativo debe de contener prácticas innovadoras para la mejora del aprendizaje.
Dicho material didáctico fue utilizado para realizar las clases más llamativas e interesantes para los estudiantes, agregando contenido multimedia que tuviera sentido con el temario de programación del ***iOS Development Lab*** utilizando la temática de la caricatura" The Powerpuff Girls" o "Las Chicas Súper Poderosas" el cual se llamó Club **PWR CODE**.

Se decidió dividir la guía de aprendizaje (Playgrounds) en los siguientes 6 capítulos. A continuación, se muestra el esquema donde está dividida la guía de aprendizaje:

- Capítulo I: Introducción a Swift con Playgrounds

  Objetivo: En el siguiente capítulo se pretende dar una bienvenida al lenguaje de programación Swift y a la herramienta tecnológica Playgrounds.

  1. ¡Bienvenida a Xcode: Playgrounds!
  2. Introducción al lenguaje de Programación Swift.
  3. Tipos de Datos.
  4. Tipos de Variables: var y let.

5. Manejo de Operadores.
6. Concatenación en sus diferentes formas.

- Capítulo II: Introducción a Solución Algorítmica de Problemas

Objetivo: Este capítulo tiene como principal objetivo dar a los alumnos las herramientas lógico-cognitivas para la resolución de problemas computacionales.

7. Solución Algorítmica de Problemas
   a) Introducción a Conceptos Básicos. (Problema, Solución, Algoritmo)
   b) Clasificación de Algoritmos. (Cualitativo, Cuantitativo)
   c) Etapas de resolución de un problema computacional.
   d) Herramientas Cognitivas:
      - Diagramas de Flujo.
      - Pseudocódigo.

- Capítulo III: Colecciones de datos

Objetivo: En este capítulo se desea introducir a los estudiantes al conocimiento de los elementos pertenecientes a las colecciones en Swift: Array, Set y Diccionarios.

8. Colecciones de datos
   a) Arrays y sus propiedades.
   b) Arrays Bidimensionales.
   c) Set.
   d) Diccionarios.

- Capítulo IV: Bucles

Objetivo: El capítulo IV pretende mostrar las formas en las que una instrucción puede repetirse una cantidad finita de veces según sea lo deseado, y para eso se demuestran diferentes formas de llevarlo a cabo.

9. Bucles
   a) For.
   b) While.
   c) Repeat – While.

- Capítulo V: Condicionales y Funciones

  Objetivo: En el siguiente capítulo se quiere llegar al tema de las condicionales, ya que son parte fundamental de los conocimientos básicos de la programación y una vez aprendidos, avanzar a lo que son las funciones y como fusionamos todo lo anteriormente aprendido dentro de las funciones.

  10. Condicionales
      a) If – Else.
      b) Else If.
      c) Switch.
  11. Funciones.

- Capítulo VI: Reforzamiento del nivel básico

  Objetivo: El objetivo de este capítulo es poner a prueba los conocimientos de los alumnos y lograr reforzar lo aprendido durante el nivel básico, y así puedan obtener notas aprobatorias para el siguiente nivel del club de programación.

  11. Hacker Rank

Cada capítulo contiene ejemplos, experimentos, tareas y actividades que son productos entregables que refuerzan las enseñanzas en el **PWR CODE** impartidas por los mentores del ***iOS Development Lab***. Adicionalmente, es importante recalcar que en el presente trabajo se quiso documentar y llevar a cabo un libro de estudio para el nivel básico a la introducción del lenguaje de programación con **Swift**, pero también se puede desarrollar el uso del lenguaje de marcado en los archivos Swift. Este tipo de archivo se utiliza en diferentes herramientas dentro de **Xcode** para una programación orientada a objetos. Y así también tener un apoyo de aprendizaje en niveles más avanzados dentro del desarrollo de aplicaciones móviles Apple.

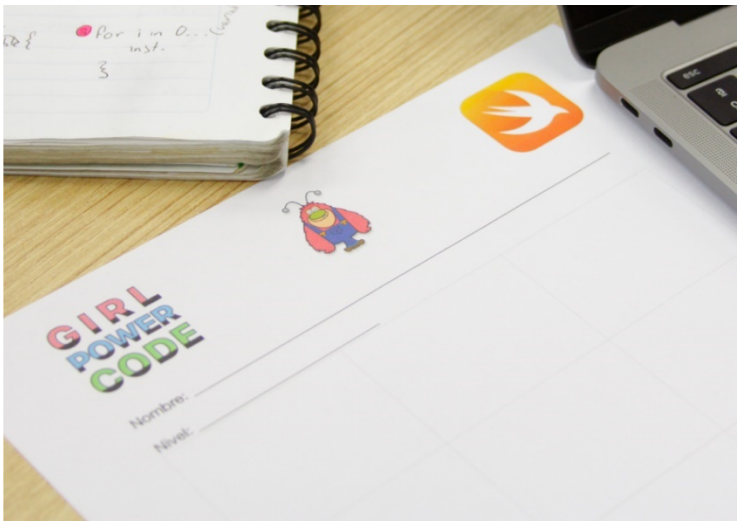
5.4.2 Tabla de material propuesto

En la *Tabla 10* se muestra las sugerencias didácticas y de evaluación que los mentores del *iOS Development Lab* pueden seguir:

Tabla 10: Sugerencias didácticas y de evaluación

Sugerencias didácticas	Sugerencias de evaluación del aprendizaje
• Utilizar la guía de aprendizaje: "Mis primeros pasos en Swift". • Utilizar tecnologías multimedia • Resolver ejercicios dentro y fuera de clase. • Aplicar el método de aprendizaje basado en problemas. • Contextualizar los problemas tanto en época como situación.	• Informes de prácticas. • Participación en clase. • Solución de ejercicios. • Trabajos y tareas. • Programar aplicaciones por computadora. • Examen Final con ayuda de Kahoot.[2]

En la *Fotografía 3* se muestra una hoja de control de tareas y trabajos. Por cada actividad realizada las alumnas obtenían sellos por consecuencia beneficios al final del curso.



Huerta, E. (2019). Hoja de control de tareas y trabajos para PWR GRL CODE. [Fotografía 3]

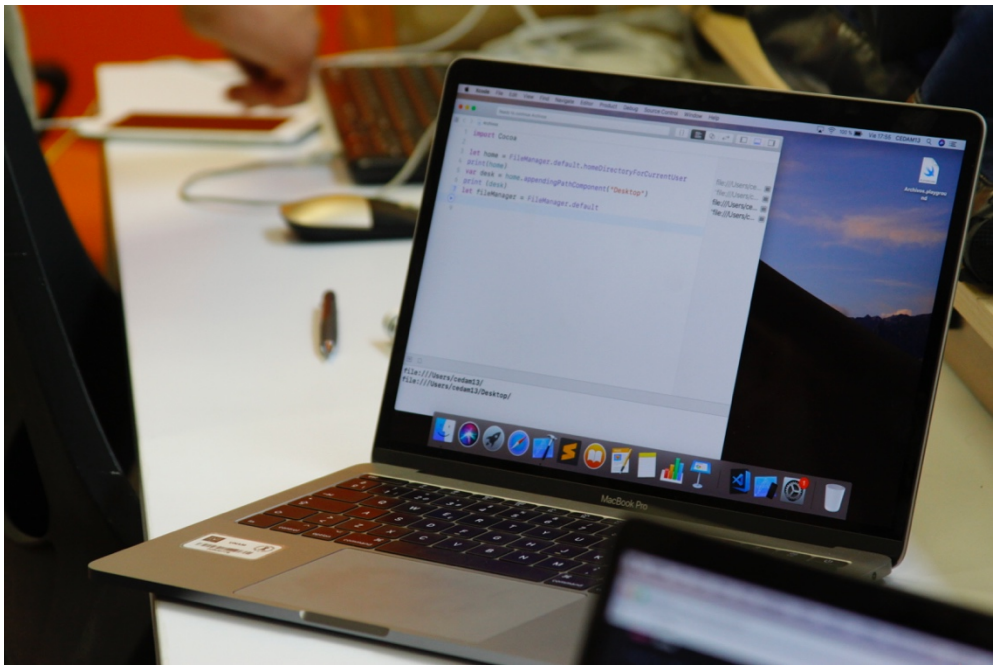
[2] Kahoot! (2020). Sitio web: https://kahoot.com

5.4.3 Beneficios de la guía de aprendizaje

El beneficio de tener la guía de aprendizaje con un temario para la enseñanza del lenguaje Swift y la SAP computacionales, es que el docente tenga una clase bien preparada y el dominio del tema y que, por consecuencia, pueda resolver las dudas de los estudiantes de una mejor forma. Con este factor, el mentor podrá poner atención a qué mejoras realizar en su clase y a tener una atención personalizada con sus alumnos. En caso de tener una actualización del lenguaje, fácilmente el mentor puede modificar en cualquier momento el contenido de la guía de aprendizaje.

En la sección de Anexos de este trabajo de tesis se podrán encontrar imágenes del contenido del Libro de Aprendizaje: **"Mis primeros pasos en Swift".**

A continuación, *Fotografía 4* se observa un equipo de cómputo perteneciente al *iOS Development Lab* de la F.E.S. Acatlán siendo utilizada para la enseñanza de la programación:



Huerta, E. (2019). Equipo de cómputo del iOS Development Lab. [Fotografía 4]

Capítulo VI. Aplicación de las fases de Scrum al proyecto PWR CODE

En el presente capítulo se muestra de forma sistemática el cómo llevar cada uno de los pasos y fases dentro de la metodología elegida, Scrum, dentro del proyecto educativo PWR CODE como una guía a seguir para las generaciones futuras dentro de la F.E.S. Acatlán. PWR CODE tiene como objetivo llevar conocimientos de programación sólidos a toda la comunidad de la F.E.S. Acatlán a través de los recursos humanos y tecnológicos del ***iOS Development Lab***. A continuación, se muestra en la *Imagen F* el Flujo que lleva la Metodología Scrum.

Romero, G. (2019). Flujo de trabajo Scrum. [Imagen F]. Platzi Recuperado de: https://platzi.com/clases/Scrum/

Como se observa, la **fase de inicio** desarrolla: el caso de negocio del proyecto del club de programación llamado PWR CODE, el proceso de la priorización del backlog del club para la obtención de los requisitos mínimos para su primera versión, p la realización de la visión de PWR CODE y el cronograma de lanzamiento para los estudiantes.

La **fase de planificación y estimación** es la siguiente en hacerse presente en la *Imagen F*, se basa en: la realización de la lista priorizada de tareas para el *Sprint correspondiente* también llamada *Sprint Backlog* y la estimación por puntos de estas tareas, en caso de ser necesario. Desde esta fase hasta la última, se realiza el proceso iterativo para encontrar mejoras en el desarrollo de los entregables.

La siguiente fase, llamada **fase de implementación**, es aquella en donde se lleva a cabo: la realización del *Sprint* correspondiente. Las tareas de la lista priorizada del *Sprint Backlog* son llevadas a cabo, dando como resultado la creación del proceso de enseñanza-aprendizaje que se lleva dentro del club PWR CODE. Igualmente, se efectúan las ceremonias diarias (*Daily Standup*) donde se da un reporte breve a todo el Equipo *Scrum* y al *Scrum Master* sobre las tareas realizadas, las tareas por realizar y si existe algún impedimento para la realización de alguna tarea. Esta última parte es de suma importancia para mantener la **transparencia, inspección y adaptación** dentro del equipo.

Como penúltima fase se encuentra la **fase de revisión y retrospectiva** en la cual se proporcionan los resultados académicos obtenidos de los estudiantes a los interesados dentro del proyecto (*Poduct Owner y Stakeholders*), validando el trabajo realizado durante el *Sprint correspondiente*. Igualmente, se da una retrospectiva del trabajo como equipo y de los resultados, es aquí donde se encuentran las áreas de oportunidad y nuevas acciones a tomar para mejorar el proceso de enseñanza en las siguientes Iteraciones-*Sprints*.

Finalmente, se encuentra la **fase de lanzamiento**. En donde se lleva a la comunidad estudiantil el producto resultante del trabajo realizado durante el proyecto. Cabe resaltar que el lanzamiento puede realizarse al finalizar la Iteración-*Sprint* 0 como una prueba “demo” o al final de la última iteración-*Sprint*, todo dependerá de los interesados en el proyecto.

A continuación, se puede ver más a detalle la aplicación de las fases al proyecto PWR CODE:

6.1 Fase de Inicio - (Sprint 0)

Se comenzará presentando la **Fase de Inicio** la cual contiene los procesos de: 1) Creación de la visión del proyecto, 2-3) Identificación del Scrum Master y los miembros del Equipo Scrum, 4) Desarrollo de Épicas, 5) Creación de la lista priorizada de pendientes del producto (*Backlog*) y, finalmente, 6) La realización de la planificación de lanzamiento. En el presente caso el producto es el club de programación PWR CODE.

6.1.1 Creación de la visión del proyecto

Este proceso consiste en describir las demandas que el proyecto busca cumplir para que, una vez hecho esto, se pueda identificar cada elemento y enfocarse en el

problema. Para la creación de dicha visión se requirió tener una reunión con los interesados en el proyecto para la definición de la visión, misión, objetivos y temario que siguió el club de programación PWR CODE. A continuación, se mostrará el Artefacto resultante del proceso de Creación de la Visión del Proyecto. El logo del club de programación fue diseñado en enero 2020 y se muestra en la *Imagen Z*:

Club de programación PWR CODE

Rangel D. (2020). Logos de PWR CODE. [Imagen Z]

Objetivos:

Despertar el interés de las ramas tecnológicas en alumnos de todas las áreas de estudio dentro de la F.E.S. Acatlán, a través del proceso de enseñanza de la **SAP** aplicado a la programación, por medio del conjunto de herramientas tecnológicas disponibles del Sistema Operativo macOS como lo es Xcode, con el uso del lenguaje de programación Swift. Se plantea lo anterior con el propósito de crear soluciones (aplicaciones móviles) a problemáticas dentro de la F.E.S. Acatlán y la sociedad, a través de los conocimientos obtenidos en PWR CODE.

Misión:

Formar universitarios de todas las áreas de estudio dentro de la F.E.S. Acatlán con conocimientos sólidos en el lenguaje de programación Swift de una manera ***divertida***, fuera de las aulas académicas. Con estrategias de aprendizaje didácticas y a lo largo de cada uno de los niveles del club de programación, se formarán alumnos competitivos en la rama de las ciencias de la computación mediante cada uno de los niveles del club de programación. De igual manera, se pretende construir en el camino agentes de cambio para las futuras generaciones dentro de la Facultad que den valor a la **equidad, desarrollo, inclusión, cultura, diversidad y respeto**.

Visión:
1.- Consolidarnos como el mejor club de programación a nivel UNAM, mediante propuestas didácticas de aprendizaje actualizadas a la par de las tecnologías.

2.- Hacer crecer el vínculo de la programación y la *SAP* con las diversas áreas dentro de la F.E.S. Acatlán, en especial, con la licenciatura de Matemáticas Aplicadas y Computación (M.A.C.).

3.- Hacer equipos de trabajo capaces de participar en torneos de programación como Hackathones.

4.- Crear interés por resolver problemáticas dentro de la F.E.S. Acatlán a través de los conocimientos obtenidos en PWR CODE.

Temario:
Dentro del club se llevan 3 niveles de aprendizaje: Burbuja, Bombón y Bellota.
Para todos los niveles es necesario haber tomado el curso de introducción al sistema operativo macOS y otros requisitos dependiendo del nivel. En la *Tabla 11* se muestra el contenido esquematizado de PWR CODE.

Tabla 11: Temario PWR CODE 2020

Nivel	Descripción	Aprendizaje
[3] 	Este nivel es para aquellas personas que comienzan su incursión en el lenguaje Swift. **Requisitos:** No se necesita pasar ninguna prueba para este nivel. **Insignia Nivel Burbuja:** Únicamente podrán obtenerla los estudiantes que aprueben el nivel con calificación aprobatoria en el examen final.	1.- Todos los temas vistos en la guía de aprendizaje “Mis primeros pasos en Swift”. 2.- Introducción a Git, GitHub. 3.- SAP. 4.- Trabajo en Equipo con repositorio remoto. (GitHub)

[3] Rangel D. (2020). Logos de PWR CODE.

4  	El siguiente nivel es para aquellas personas que tengan las bases básicas del lenguaje de programación Swift y comiencen con la etapa intermedia. **Requisitos:** Haber obtenido la *Insignia Nivel Burbuja* o haber aprobado un examen de conocimientos en el lenguaje de programación y la solución algorítmica de problemas. **Insignia Nivel Bombón:** Únicamente podrán obtenerla los estudiantes que entreguen el proyecto final propuesto por sus mentores y este obtenga una calificación aprobatoria.	1.- Introducción a Apps, SwiftUI. 2.- Clase, Herencia y Estructuras. 3.- Objetos: UIButton, Label, TextField, TextView, UIImage, Constrains, UISwitch, UISegmentControl, UISlider, UIPicker View. 4.- Navegation Controller, Segue, Paso por referencia. 5.- Audio y Animaciones 6.- ¿Cómo subir mi aplicación a la App Store?
5  	Este nivel es para los expertos de los tópicos de los dos niveles anteriores **Requisitos:** Haber obtenido la *Insignia Nivel Bombón* o haber aprobado un examen de conocimientos en el lenguaje de programación orientado a objetos y la solución algorítmica de problemas. **Insignia Nivel Bellota:** Únicamente podrán obtenerla los estudiantes que entreguen el proyecto final propuesto por sus mentores y este obtenga una calificación aprobatoria.	1.- Tipos de Páginas: Tab Bar, Page Control, UIScroll View. 2.- API, Framework, Librería, SDK. 3.- CocoaPods. 4.- Bases de Datos (Firebase). 5.- URL. 6.- Introducción a ARKIT.

[5] Rangel, D. (2020). Logos de PWR CODE.

Finalmente, se puede concluir lo siguiente. El club imparte desde temas básicos hasta avanzados, aprovechando todos los recursos tecnológicos que ofrece el Centro de Desarrollo de Aplicaciones Móviles – ***iOS Development Lab*** que hay en la institución. Con el objetivo de que los estudiantes desarrollen Aplicaciones Móviles. Cabe destacar que el temario del Nivel Burbuja en su mayoría se encuentra en la guía de aprendizaje: “Mis primeros pasos en Swift” previamente descrita aquí, en donde el mentor podrá actualizar los temas sí se llegase a tener una actualización nueva en el lenguaje de programación. Para los niveles posteriores se da una propuesta de temas, igualmente el mentor es libre de agregar o modificar el orden de enseñanza de algún tópico.

A continuación, se muestra en la *Imagen AA* las insignias digitales de cada nivel de programación que se les otorgará a los alumnos aprobados en los cursos.

Rangel D. (2020). Insignias PWR CODE 2020. [Imagen AA]

6.1.2 Identificación del Equipo Scrum

En este proceso se consigue al capital humano que estará efectuando los cargos de Scrum Master, los socios del proyecto y los roles centrales del equipo.
Los puestos se repartieron de la siguiente manera:

- *Stakeholder*(s)
    - Lic. Christian Carlos Delgado Elizondo, Coordinador del Programa de M.A.C. de la F.E.S. Acatlán.
    - Teotl Q. Macias Bravo, Program Manager de Apple Education.
- *Producto Owner*
    - Dr. Eduardo Eloy Loza Pacheco, Manager del *iOS Development Lab* de la F.E.S. Acatlán.

- *Scrum Master*
    - Encargado/a de PWR CODE (mentor/a del *iOS Development Lab*)
- Equipo *Scrum*
    - Mentores del *iOS Development Lab* que estén realizando su servicio social en el periodo escolar correspondiente.

6.1.3 Desarrollo de Épicas

Una vez definidos los roles, es necesario que el *Product Owner* se encargue del desarrollo de Épicas. Las épicas son historias de usuario no desarrolladas o, en otras palabras, bloques con tareas más pequeñas. Estos bloques serán desarrollados y priorizados en el siguiente proceso. A continuación, se presentan las Épicas encontradas en el proyecto de PWR CODE:

- Organización de grupos y horarios de los 3 niveles de PWR CODE.
- Preparación de clases que se impartirán en los 3 niveles de PWR CODE.
- Lanzamiento de convocatoria de inscripción.
- Reclutamiento de mentores del *iOS Development Lab*.
- Aplicación de Examen de conocimientos generales para niveles Bombón y Bellota.
- Aceptación de los estudiantes a los niveles de programación.
- Entrega de insignias a los estudiantes aprobados en los respectivos niveles.
- Apertura del club de programación PWR CODE.
- Cierre del club de programación PWR CODE.
- Evaluaciones a los estudiantes de cada nivel.

6.1.4 Creación del *Product Backlog*

En este proceso se priorizan las épicas para crear una lista priorizada de pendientes del producto y se establecen los criterios de terminado. Este trabajo lo sigue realizando el experto en el negocio: *Product Owner*, en el caso de estudio el *iOS Development Lab* Management. A continuación, se da la lista priorizada ya trabajada, ahora denominada *Product Backlog*:

1. Reclutamiento de mentores del *iOS Development Lab*.
2. Organización de grupos y horarios de los 3 niveles de PWR CODE.
3. Preparación de clases que se impartirán en los 3 niveles de PWR CODE.
4. Lanzamiento de convocatoria de inscripción.
5. Aplicación de examen de conocimientos generales para niveles Bombón y Bellota.
6. Aceptación de los estudiantes a los niveles del club.

7. Apertura del club de programación PWR CODE.
8. Evaluaciones a los estudiantes de cada nivel.
9. Cierre del club de programación PWR CODE.
10. Entrega de insignias a los estudiantes aprobados en los respectivos niveles.

Los criterios de terminación de las tareas los define el *Product Owner*, bajo la aceptación de los criterios de evaluación y verificación propuestas por este integrante del equipo *Scrum*. Él deberá de tener una sesión de revisión del sprint para dar por terminadas las tareas del *Sprint Backlog* o anunciar que hay deuda técnica en el sprint.

6.1.5 Planificación de Lanzamiento

En este proceso se realiza una reunión de revisión en donde todos los interesados del proyecto revisan las épicas para realizar un cronograma de planificación del lanzamiento en su versión uno del producto. En el club de programación se realizará una reunión con los *Stakeholders* cada inicio de semestre, para acordar las mejores fechas de apertura de PWR CODE. El resultado de esta reunión deberá ser un entregable en forma de minuta, con la fecha de inicio de trabajo del club de programación, así como una fecha de lanzamiento de la primera generación de PWR CODE en el semestre.

A continuación, se muestra un ejemplo de la minuta:

Ejemplo VI: Minuta de planificación de lanzamiento

Minuta de planificación de lanzamiento

Fecha: día/mes/año
Responsable de la minuta: Nombre Completo

Puntos por tratar:

- *Product Backlog*.
- Fecha de inicio de trabajo de PWR CODE (*Sprint* 1).
- Fecha de lanzamiento de clases del club PWR CODE.

Acciones Pendientes:

6.2 Fase de planificación y estimación - (Sprint 0)

A continuación, se presenta la **Fase de planificación y estimación** la cual contiene los procesos de: 7) Creación de historias de usuario, 8) Estimación de historias de usuario, 9) Comprometer historias de usuario, 10-11) Identificación y estimación de tareas 12) Planificación del *Sprint* 1 (*Sprint Backlog*).

6.2.1 Creación de historias de usuario

En este proceso se crean las historias de usuario, que son épicas desarrolladas. Éstas están diseñadas para asegurar que los requisitos del cliente estén claramente representados y finalmente sean incorporadas al *Product Backlog*. Las historias de usuario definen una funcionalidad, pues en una sola frase debe quedar claro **QUIEN** (rol) hará una **ACCIÓN** (objetivo) para satisfacer una **NECESIDAD** (motivación). Ejemplo: "Yo **como** [rol del usuario] **quiero** [Objetivo] **para poder** [beneficio]".
En el proyecto se detectaron las siguientes épicas y se priorizaron, pero cada una de ellas son un bloque para desarrollar. A continuación, se muestran las épicas desarrolladas y transformadas en historias de usuario:

1. Reclutamiento de mentores del *iOS Development Lab*.

👥Rol del Usuario: Manager del *iOS Development Lab – Product Owner.*
🎯Objetivo: Reclutar mentores capacitados ampliamente que puedan resolver dudas durante las clases de PWR CODE.
🎖️Beneficio: Estudiantes mejor preparados siendo impulsados por sus mentores.

Historia de Usuario: Yo **como** Manager del *iOS Development Lab* **quiero** reclutar mentores capacitados ampliamente que puedan resolver dudas durante las clases de PWR CODE **para poder** tener estudiantes mejor preparados siendo impulsados por sus mentores.

2. Organización de grupos y horarios de los 3 niveles de PWR CODE.

👥Rol del Usuario: Manager del *iOS Development Lab – Product Owner* y *Stakeholders.*
🎯Objetivo: Fijar días, horarios y grupos a cada nivel del club de programación.
🎖️Beneficio: Tener horarios disponibles y accesibles tanto para los estudiantes como para los mentores sin interferir en los horarios de clase de materias curriculares dentro del *iOS Development Lab*.

Historia de Usuario: Yo **como** Manager del *iOS Development Lab* y *Stakeholder* **quiero** fijar días, horarios y grupos a cada nivel del club de programación **para poder** tener horarios disponibles y accesibles tanto para los estudiantes como para los mentores sin interferir en los horarios de clase de materias curriculares dentro del *iOS Development Lab*.

3. Preparación de clases que se impartirán en los 3 niveles de PWR CODE.

👥Rol del Usuario: Mentores del *iOS Development Lab* – Equipo *Scrum.*
🎯Objetivo: Preparar material para los estudiantes inscritos en los respectivos niveles de programación.
🎖️Beneficio: Mentores altamente preparados para impartir los temas de cada nivel y resolver dudas en clase.

Historia de Usuario: Nosotros **como** Mentores del *iOS Development Lab* **queremos** preparar material didáctico **para poder** estar altamente preparados para impartir los temas de cada nivel y resolver dudas en clase.

4. Lanzamiento de convocatoria de inscripción.

👥Rol del Usuario: Encargado/a de PWR CODE - *Scrum Master.*
🎯Objetivo: Lanzar la convocatoria de inscripción a los estudiantes hacia todos los niveles del club de programación.
🎖️Beneficio: Obtener un control de inscripciones, una lista con toda la información escolar de los alumnos y un censo de popularidad entre la comunidad estudiantil hacia el interés del club de programación. Que permita tener una medición para futuras mejoras.

Historia de Usuario: Yo **como** Encargado/a de PWR CODE **quiero** lanzar la convocatoria de inscripción a los estudiantes hacia todos los niveles del club de programación **para poder** obtener un control de inscripciones, una lista con toda la información escolar de los alumnos y un censo de popularidad entre la comunidad estudiantil hacia el interés del club de programación. Que permita tener una medición para futuras mejoras.

5. Aplicación de examen de conocimientos generales para niveles Bombón y Bellota.

👥Rol del Usuario: Mentores del *iOS Development Lab* en turno – Equipo *Scrum.*
🎯Objetivo: Realizar un examen de conocimientos generales para estudiantes que no cursaron el nivel de programación anterior al que aplican.
🎖️Beneficio: Localizar el nivel de programación de los estudiantes y capacitarlos en el nivel adecuado dentro de PWR CODE.

Historia de Usuario: Nosotros **como** Mentores del *iOS Development Lab* **queremos** realizar un examen de conocimientos generales para estudiantes que no cursaron el nivel de programación anterior al que aplican **para poder** localizar el nivel de programación de los estudiantes y capacitarlos en el nivel adecuado dentro de PWR CODE.

6. Aceptación de los estudiantes a los niveles del club.

👥Rol del Usuario: Mentores del *iOS Development Lab* en turno – Equipo *Scrum.*
🎯Objetivo: Mandar correos de aceptación a los estudiantes que alcanzaron cupo en el nivel, grupo y horario deseado de PWR CODE.
🎖️Beneficio: Informar a los estudiantes aceptados y una lista de integrantes de cada grupo y horario para pasar asistencia durante la duración del club de programación.

Historia de Usuario: Nosotros **como** Mentores del *iOS Development Lab* **queremos** mandar correos de aceptación a los estudiantes que alcanzaron cupo en el nivel, grupo y horario deseado de PWR CODE **para poder** informar a los estudiantes aceptados y una lista de integrantes de cada grupo y horario para pasar asistencia durante la duración del club de programación.

7. Apertura del club de programación PWR CODE.

👥Rol del Usuario: Encargado/a de PWR CODE - *Scrum Master.*
🎯Objetivo: Publicar fechas de apertura a las clases del club de programación en las redes sociales.
🎖️Beneficio: Informar a los estudiantes sobre las fechas de inicio de las clases del club de programación.

Historia de Usuario: Yo **como** Encargado/a de PWR CODE **quiero** publicar fechas de apertura a las clases del club de programación en las redes sociales **para poder**

informar a los estudiantes sobre las fechas de inicio de las clases del club de programación.

8. Evaluaciones a los estudiantes de cada nivel.

👥Rol del Usuario: Mentores del *iOS Development Lab* en turno – Equipo *Scrum.*
🎯Objetivo: Evaluar a los estudiantes al finalizar las clases del nivel donde se encuentren inscritos.
🎖️Beneficio: Apoyar las áreas de oportunidad de los estudiantes no aprobados.

Historia de Usuario: Nosotros **como** Mentores del *iOS Development Lab* **queremos** evaluar a los estudiantes al finalizar las clases del nivel donde se encuentren inscritos **para poder** apoyar las áreas de oportunidad de los estudiantes no aprobados.

9. Cierre del club de programación PWR CODE.

👥Rol del Usuario: Encargado/a de PWR CODE - *Scrum Master.*
🎯Objetivo: Realizar una ceremonia de cierre con las autoridades institucionales correspondientes.
🎖️Beneficio: Dar cierre a las clases del club de programación.

Historia de Usuario: Yo **como** Encargado/a de PWR CODE **quiero** realizar una ceremonia de cierre con las autoridades institucionales correspondientes **para poder** dar cierre a las clases del club de programación.

10. Entrega de insignias a los estudiantes aprobados en los respectivos niveles.

👥Rol del Usuario: Encargado/a de PWR CODE - *Scrum Master.*
🎯Objetivo: Otorgar las insignias de cada nivel del club de programación a los estudiantes aprobados.
🎖️Beneficio: Reconocer a los estudiantes mejor preparados que pertenecerán a los siguientes niveles del club de programación.

Historia de Usuario: Yo **como** Encargado/a de PWR CODE **quiero** otorgar las insignias de cada nivel del club de programación a los estudiantes aprobados **para poder** reconocer a los estudiantes mejor preparados que pertenecerán a los siguientes niveles del club de programación.

6.2.2 Estimación de historias de usuario

Una vez terminadas las historias de usuario, el *Scrum Master* se encarga de realizar una reunión para la estimación de esfuerzo de las Historias de usuario con el equipo de desarrollo. Aquí es donde se hace uso del *Planning Poker*. *Planning Poker* es una técnica para estimar el esfuerzo y/o el tamaño relativo de las tareas de desarrollo. La opción de estimación elegida trata de puntos basados en la serie Fibonacci, ya que su gran diferencia entre el primer número y el posterior denotan una diferencia de complejidad al momento de asignarlo en las tareas.
El *Scrum Master* y el Equipo *Scrum* deben leer y comprender cada una de las historias de usuario. Una vez realizado esto, cada miembro debe asignarle un número dentro del *Planning Poker* y mostrarlo todos al mismo tiempo.
Habrá miembros del equipo que tendrán la misma puntuación, pero las personas que tengan una estimación más baja o alta a la popular podrán luchar por subir o bajar la puntuación asignada a la tarea con opiniones técnicas. Finalmente, ya escuchados todos los puntos de vista, se volverá a tener una votación que será decisiva y final. Para exponer las asignaciones del *Planning Poker* pueden usarse desde tarjetas físicas con la serie Fibonacci, las manos o aplicaciones móviles como lo es Planning Poker MX.

En la *Imagen BB* se muestra la aplicación disponible en la tienda App Store, igualmente se encuentra en Play Store:

App Store. (2020). Aplicación Planning Poker MX. [Imagen BB]. Recuperado de: https://play.google.com/store/apps/details?id=banana.planningpoker&hl=es_MX

En la aplicación móvil se pueden observar varias opciones para trabajar, la *Imagen CC* muestra la ya elegida serie Fibonacci, la *Imagen DD* hace referencia a la técnica

del tamaño de tallas en playeras y la *Imagen EE* es una versión personalizada en donde podemos elegir el valor de las tarjetas las cuales tienen asignadas a personajes para hacer la estimación más divertida.

Imagen CC: Técnica de Fibonacci

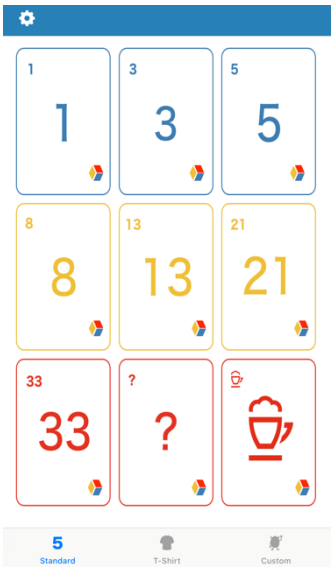


Imagen DD: Técnica de tallas

Imagen EE: Versión personalizada

La tarjeta de “Taza de café” representa una asignación desconocida. Cuando un integrante del equipo está dudoso de que asignación darle por falta de conocimientos técnicos u otra situación de esta índole, puede elegir esta tarjeta que se muestra en la *Imagen FF* a continuación:

Imagen FF: Tarjeta “Taza de café”

6.2.3 Comprometer historias de usuario

En este proceso, una vez que las historias de usuario ya se encuentran estimadas, el equipo de desarrollo se debe comprometer y hacer responsable de las historias de usuario que decidan asignarse, como se muestra en la *Imagen GG*. Como un Equipo *Scrum*, la asignación de las historias de usuario debe ser auto organizada por los miembros del equipo.

Anónimo. (2018). Equipo auto-organizado. [Imagen GG]. Recuperado de: https://viewnext.usal.es/blog/principios

6.2.4 Identificación y estimación de tareas

Las historias de usuario probadas, estimadas y asignadas se dividen en tareas específicas y se compilan en una lista de tareas. Durante la reunión de planificación de tareas, el Equipo de *Scrum* estima el esfuerzo necesario para realizar cada tarea en la lista. La estimación tiene doble función dentro de las tareas: identificar complejidad, pero también dar a conocer el tiempo que se tomará en realizarlas durante un *Sprint*.
A continuación, se mostrará la lista de tareas encontradas en cada bloque de historias de usuario:

1. Reclutamiento de mentores del *iOS Development Lab*.
    - 1.1 Lanzar convocatoria para pertenecer a los mentores del club PWR CODE.
    - 1.2 Realizar entrevistas a los postulantes.
    - 1.3 Seleccionar a los mentores más capacitados para ser parte de PWR CODE.
    - 1.4 Informar a los seleccionados sobre su participación en el club de programación.

2. Organización de grupos y horarios de los 3 niveles de PWR CODE.
    2.1 Realizar junta de planificación de lanzamiento del club de programación.
        2.1.2 Fijar días, horarios y grupos para cada nivel de PWR CODE.
        2.1.3 Asignar grupos a los mentores que se harán responsables de impartir las clases en el club de programación.

3. Preparación de clases que se impartirán en los 3 niveles de PWR CODE.
    3.1 Realizar la preparación del material que se utilizará en las clases de nivel Burbuja.
    3.2 Realizar la preparación del material que se utilizará en las clases de nivel Bombón.
    3.3 Realizar la preparación del material que se utilizará en las clases de nivel Bellota.

4. Lanzamiento de convocatoria de inscripción.
    4.1 Realizar formulario de inscripción para los distintos niveles del club de programación.
    4.2 Lanzar convocatoria en redes sociales.
    4.3 Publicar en redes sociales horarios y grupos de cada nivel de PWR CODE.

5. Aplicación de examen de conocimientos generales para niveles Bombón y Bellota.
    5.1 Realizar examen de conocimientos generales para nivel Bombón.
    5.2 Realizar examen de conocimientos generales para nivel Bellota.
    5.3 Realizar formulario para aplicación de examen de conocimientos.
    5.4 Lanzar formulario a redes sociales.
    5.5 Fijar fecha, hora y lugar para aplicación de exámenes.
    5.6 Enviar correos de asignación de nivel a los estudiantes aprobados.

6. Aceptación de los estudiantes a los niveles del club.
    6.1 Enviar correos de aceptación a los estudiantes que no realizaron examen de conocimientos.

7. Apertura del club de programación PWR CODE.
    7.1 Iniciar clases de programación de todos los niveles de PWR CODE.
    7.2 Pasar lista en los grupos.

8. Evaluaciones a los estudiantes de cada nivel.
    8.1 Preparar y aplicar evaluaciones para nivel Burbuja.
        8.1.2 Entregar resultados de evaluación.
        8.1.3 Realización de insignias para estudiantes aprobados.
    8.2 Preparar y aplicar evaluaciones para nivel Bombón.
        8.2.2 Entregar resultados de evaluación.
        8.2.3 Realización de insignias para estudiantes aprobados.

    8.3 Preparar y aplicar evaluaciones para nivel Bellota.
        8.3.2 Entregar resultados de evaluación.
        8.3.3 Realización de insignias para estudiantes aprobados.

9. Cierre del club de programación PWR CODE.
    9.1 Fijar fecha y hora para la clausura.
    9.2 Invitar a las autoridades escolares para entrega de insignias.
    9.3 Publicar en redes sociales la fecha y hora de clausura.
    9.4 Informar a los estudiantes aprobados sobre la entrega de insignias.

10. Entrega de insignias a los estudiantes aprobados en los respectivos niveles.
    10.1 Preparar el laboratorio para la ceremonia.
    10.2 Tener equipo de grabación para publicar en redes sociales la ceremonia.
    10.3 Realizar un horario de entrega de insignias.

6.2.5 Planificación de Sprint 1 (*Sprint Planning*)

En este proceso, el *Scrum Master* y el Equipo de *Scrum* llevan a cabo una reunión de planificación del *Sprint*. El *Sprint* es un bloque de tiempo de un mes o menos durante el cual se crea un incremento del producto “terminado” y se cumple la meta del *Sprint* (*Sprint Goal)* propuesta desde el inicio de éste.

Definición del Sprint Goal

El objetivo del *Sprint* 1 del club de programación será: **Tener listos los preparativos para el inicio de clases de PWR CODE.**

Realización del Sprint Backlog 1

Conforme a la estimación de historias de usuario, el *Scrum Master* puede comprometer ciertas tareas del *backlog* para ser realizadas en el primer *Sprint,* dependiendo de la duración de éste. Aquí es de suma importancia que el Equipo de

Scrum conozca sus capacidades y velocidades de trabajo para realizar tareas con cierto grado de complejidad en un determinado tiempo, ya que de eso dependerá el éxito del *Sprint 1.*

En el *Ejemplo VII* se propondrá un caso prueba del *Sprint Backlog* con una suma total de tareas sólo para facilitar la comprensión de los resultados de este proceso.

Ejemplo VII: Caso de prueba de Sprint Backlog

Cada *Sprint* tendrá una duración de 4 semanas, los puntos al final de cada línea son los puntos estimados para cada tarea.

Sprint Backlog:

- ✓ Lanzar convocatoria para pertenecer a los mentores del club PWR CODE. (3)
- ✓ Realizar entrevistas a los postulantes. (8)
- ✓ Seleccionar a los mentores más capacitados para ser parte de PWR CODE. (5)
- ✓ Informar a los seleccionados sobre su participación en el club de programación. (3)
- ✓ Realizar junta de planificación de lanzamiento del club de programación. (3)
- ✓ Fijar días, horarios y grupos para cada nivel de PWR CODE. (8)
- ✓ Asignar grupos a los mentores que se harán responsables de impartir las clases en el club de programación. (5)
- ✓ Realizar la preparación del material que se utilizará en las clases de cada nivel del club de programación. (13)
- ✓ Realizar formulario de inscripción para los distintos niveles del club de programación. (5)
- ✓ Lanzar convocatoria en redes sociales. (3)
- ✓ Publicar en redes sociales horarios y grupos de cada nivel de PWR CODE. (5)
- ✓ Realizar examen de conocimientos generales para nivel Bombón. (8)
- ✓ Realizar examen de conocimientos generales para nivel Bellota. (8)
- ✓ Realizar formulario para aplicación de examen de conocimientos. (5)
- ✓ Lanzar formulario a redes sociales. (3)
- ✓ Fijar fecha, hora y lugar para aplicación de exámenes. (3)
- ✓ Enviar correos de asignación de nivel a los estudiantes aprobados que realizaron examen de conocimientos generales. (5)

- ✓ Enviar correos de aceptación al club de programación a los estudiantes que no realizaron examen de conocimientos. Para alumnos que no lo requieren. (5)

Total, de puntos estimados para el *Sprint 1 = 106 puntos*

6.3 Fase de implementación y lanzamiento - (Sprint 1)

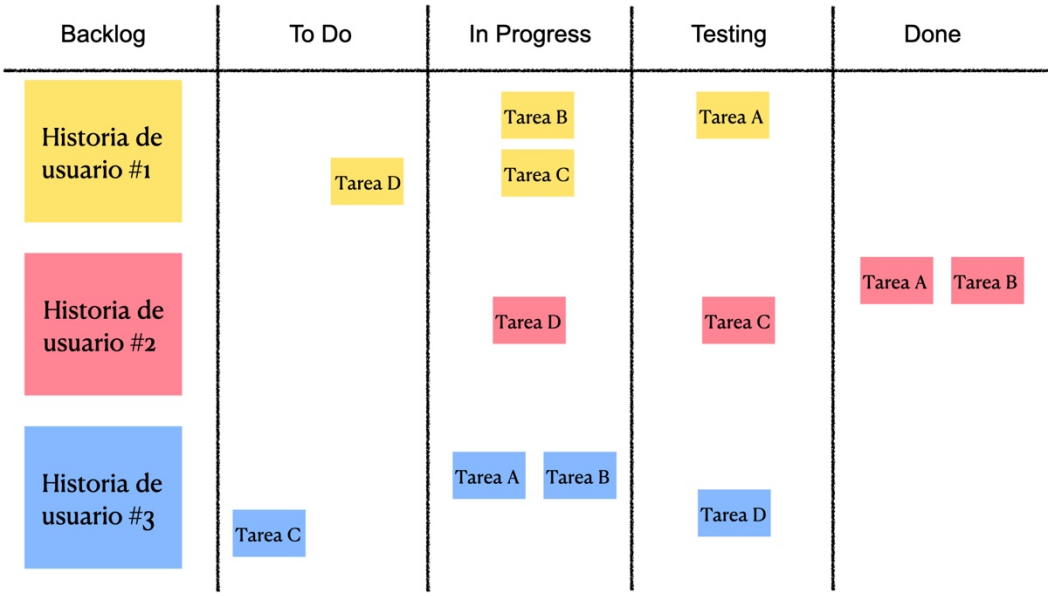
A continuación, se presenta la **Fase de implementación y lanzamiento** la cual contiene los procesos de: 13) Creación de entregables, 14) Realización del *Daily Standup* y 15) Refinamiento del *Backlog*.

6.3.1 Creación de entregables y tablero Scrum (Scrum Board)

El equipo de *Scrum* trabajará en el *Sprint Backlog* para crear los entregables. Generalmente se utiliza un tablero de *Scrum (Scrum Board)* para dar seguimiento a las tareas que se llevan a cabo y tener transparencia durante el proceso de creación. Este tablero es de suma importancia para todo el Equipo *Scrum.*

En la siguiente *Imagen HH,* se muestra un *Scrum Board* en proceso de trabajo. Como se puede observar, está dividido en 5 secciones. La primera es "*Backlog",* en la que se situarán todas las historias de usuario del proyecto. En la segunda, denominada "*To Do*", se encontrará la lista de tareas a realizar de cada historia de usuario. En la *Imagen HH* se visualiza que las tareas correspondientes a la "Historia de usuario #1" se encuentran de color amarillo así se puede ver el avance de este bloque de una mejor manera. Las tareas una vez que son tomadas para ser realizadas por los integrantes del Equipo *Scrum* deben de ser pasadas a la siguiente columna llamada "*In Progress*", esto indicará que la tarea ya está en proceso de realización. Ya terminada la tarea, es momento de que en el *Scrum Board* sea pasada a la siguiente columna nombrada "*Testing*". Esta cuarta columna señala que se está verificando que la tarea realizada cumpla con las condiciones de "terminado". Si logra cumplir con todos los requisitos, esta tarea puede pasar a la columna final "*Done*", la cual indica que se ha cerrado el proceso de elaboración de la tarea.

Imagen HH: Ejemplo de un Scrum Board



Conforme se va avanzando en el *Sprint,* se va creando el diagrama de *Burndown Chart*, el cuál es una representación gráfica del trabajo realizado dentro de un proyecto en un periodo de tiempo. Este diagrama es útil para predecir si realmente se logrará terminar a tiempo todo el trabajo asignado durante el *Sprint 1*.

6.3.2 Diagrama Burndown Chart

A continuación, en el *Ejemplo VIII* se mostrará el seguimiento del caso prueba del *Ejemplo VI* para la realización del diagrama *Burn Down Chart* y así lograr una mejor comprensión de su realización:

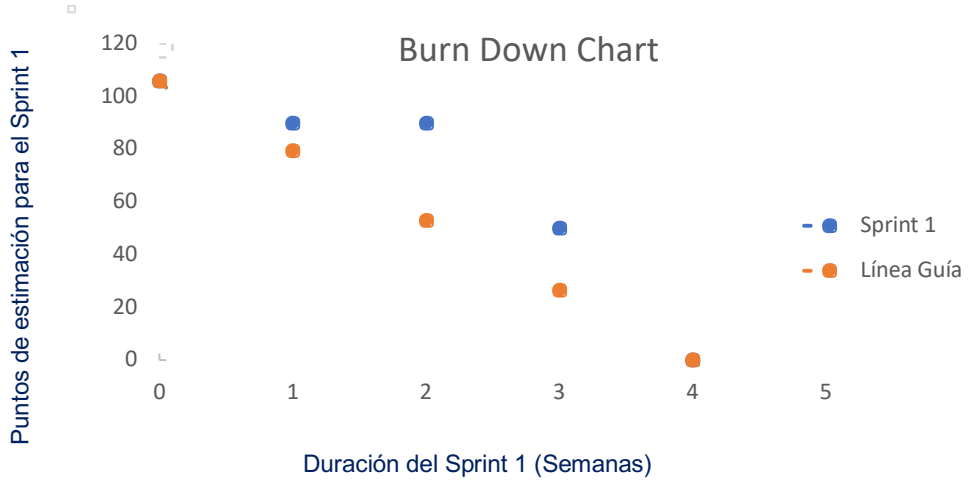
Ejemplo VIII: Diagrama Burndown chart del caso de prueba

Como se mostró en el *Ejemplo VI*, se tomará cada *Sprint* con una duración de 4 semanas, y para el *Sprint* 1 se propuso un total de 106 puntos.
En la *Gráfica A,* el eje horizontal representa las semanas que durará el *Sprint 1* y el eje vertical representa los puntos de estimación asignados. La línea naranja es la "Línea Guía". Esta línea indicará el trabajo continuo que se debería seguir para cumplir todas las tareas dentro del tiempo de duración del *Sprint*. Los puntos totales son 106 y hay un tiempo de trabajo de 4 semanas, haciendo los cálculos necesarios debería de trabajarse un promedio de 26.5 puntos de estimación por semana para lograr el objetivo. Sin embargo, no siempre suele ser así la forma en la que se realizan las tareas, todo depende del ritmo de trabajo del Equipo *Scrum.* En este caso prueba el Equipo *Scrum* logró en la primera semana restar 16 puntos, lo cual

no se alejaba mucho de la “Línea Guía” como se muestra en la línea azul. La segunda semana no se cerró ninguna tarea, por lo tanto, la gráfica se mantuvo continua y es aquí donde se observa el posible riesgo de no lograr los objetivos del *Sprint*. La tercera semana el equipo cerró tareas con 40 puntos, esto mejoraba las expectativas del *Sprint.* Finalmente, se puede apreciar que el equipo tuvo una sobrecarga de trabajo en la semana previa a la entrega, pero lograron terminar todas las tareas.

Es así como el diagrama *Burndown Chart* ayuda a la estimación del trabajo.

Gráfica A: Burndown Chart



6.3.3 Realización del Scrum diario (Daily Standup)

Este proceso se lleva a cabo diariamente con una reunión altamente focalizada durante un bloque de tiempo asignado de 15 minutos llamada *Scrum* Diario *(Daily Standup)* entre el S*crum Master* y los Integrantes del Equipo *Scrum.* Es un foro para que el equipo se ponga al día sobre sus progresos y obstáculos *(Blockers)* que puedan presentar y se requiera la ayuda del *Scrum Master*.
En este proyecto de investigación la persona encargada de PWR CODE (mentor/a del *iOS Development Lab*), que funge el rol de *Scrum Master*, será quien dirija esta reunión y quien resuelva los posibles obstáculos (*Blockers*) que comunique el Equipo de *Scrum* conformado por los mentores del *iOS Development Lab*.

En la *Fotografía 5* se muestra al equipo de mentores y autoridades escolares del *iOS Development Lab* generación 2019-2:

Fotografía 5: Mentores y autoridades escolares del iOS Development Lab y la F.E.S Acatlán

6.3.4 Mantenimiento de la lista priorizada de pendientes (*Backlog Refinement*)

Constantemente se debe actualizar y dar mantenimiento al *Product Backlog* del proyecto. Scrum es una metodología ágil que tendrá como consecuencia la creación de nuevas tareas que no se habían propuesto desde el *Sprint 0,* por lo tanto, es tarea del *Scrum Master* mantener el *Scrum Board* actualizado y realizar las ceremonias necesarias para la estimación de las nuevas tareas con el Equipo *Scrum.*

6.4 Fase de revisión y retrospectiva - (Sprint 1)

A continuación, se presenta la **Fase de revisión y retrospectiva** la cual contiene los procesos de: 16) Demostrar y Validar el *Sprint 1* y 17) Retrospectiva del *Sprint* 1.

6.4.1 Revisión de Sprint (*Sprint Review*)

En este proceso el Equipo de *Scrum* muestra los entregables del *Sprint* 1 al *Product Owner.* En el caso de esta investigación, los resultados son mostrados al manager del *iOS Development Lab* de la F.E.S. Acatlán en una ceremonia llamada Revisión de *Sprint (Sprint Review).* El propósito de esta reunión es lograr la aprobación y aceptación respecto al producto o servicio resultante del *Sprint 1*. Igualmente es

para escuchar nuevas mejoras al proceso ya realizado, llegando así a un proceso iterativo, lo que es el objetivo de la metodología a cargo del proyecto educativo.

6.4.2 Retrospectiva de Sprint (*Sprint Retrospective*)

En este proceso el *Scrum Master* se encarga de realizar una ceremonia llamada Retrospectiva de *Sprint* (*Sprint Retrospective*). En esta ceremonia, el *Scrum Master* y el equipo de desarrollo se reúnen para discutir las lecciones aprendidas durante el *Sprint 1*. Dicha información se documenta como mejoras accionables aceptadas llamadas *"Agreed Actionable Improvements"* que servirán para el siguiente Sprint y que cada integrante del Equipo Scrum se lleva como aprendizaje, y así, lograr un crecimiento de desarrollo como equipo, dando paso a la **comunicación, transparencia y adaptación.**

6.5 Fase de lanzamiento - (*Sprint* 1)

A continuación, se presenta la **Fase de lanzamiento** la cual contiene los procesos de: 16) Enviar entregables y 17) Retrospectiva del Proyecto.

6.5.1 Enviar entregables

En este proceso se hace énfasis en la entrega al cliente de los entregables aceptados. En el caso de estudio que se lleva en esta investigación, los resultados de los entregables se presentan a los *Stakeholder*s del proyecto como lo son: el Coordinador del Programa de M.A.C. de la F.E.S. Acatlán (Lic. Christian Carlos Delgado Elizondo) y al *Program Manager* de Apple *Education* (Teotl Q. Macías Bravo). La conclusión satisfactoria del *Sprint 1* se documenta en un acuerdo formal de entregables funcionales.

6.5.2 Retrospectiva del proyecto

En este proceso final se concluye el proyecto. Los socios de la organización y los miembros del equipo principal de *Scrum* se reúnen para hacer una retrospectiva del proyecto, identificar y documentar las lecciones que se aprendieron, para mejorar futuras iteraciones. En el proyecto PWR CODE, al finalizar la ceremonia de clausura del club de programación, es necesario ver los resultados finales de los conocimientos de los alumnos, detectar áreas de oportunidad y realizar estas mejoras en las iteraciones posteriores siguiendo todo el proceso ya anteriormente descrito en esta investigación durante el Capítulo VI.

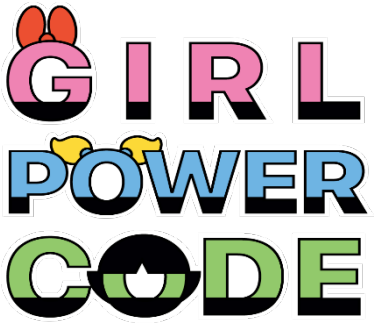
Capítulo VII. Resultados del proyecto (2019-2/2020-1) y nuevas propuestas

7.1 Resultados del proyecto (2019-2/2020-1)

La implementación se llevó acabo en el laboratorio *iOS Development Lab* de la **F.E.S. Acatlán**. Se tomó como primer "piloto" un grupo femenino de la licenciatura de **M.A.C.**, debido a que se deseaba aumentar el interés de la programación en ellas. El club de programación se nombró **PWR GRL CODE** para la generación 2019-2. Una de las principales prioridades en esta generación fue la creación pronta de los requerimientos mínimos para la formalización del club de programación como lo fue: la definición de los niveles técnicos de aprendizaje, temarios que incluyeran tópicos esenciales de SAP y el lenguaje de programación Swift; y actividades basadas en la temática de una caricatura infantil llamada "*The Powerpuff Girls*" con el fin de obtener el interés de las estudiantes. Esta fue la **primera iteración** del proyecto, se conformó del *Sprint* 0 y *Sprint* 1.
Los resultados del club de programación se pudieron observar al final de cada semestre en las evaluaciones académicas finales o en los resultados obtenidos de los estudiantes que decidieron participar en diversos eventos de programación de la F.E.S. Acatlán, como lo es el Hackathon, organizado por el ***iOS Development Lab*** y la coordinación de **M.A.C.**

A continuación, se mostrará el primer documento de requerimientos mínimos para la formalización del club de programación, que surgió como entregable a lo largo de la Iteración 1. Se incluyen los objetivos, misión, visión, un temario y en la *Imagen II* se muestra el logo del club de programación **PWR GRL CODE**.

Club de Programación PWR GRL CODE 2019-2



Contreras, A. (2019). Logo de PWR GRL CODE 2019-2. [Imagen II]

Objetivos:
Integrar a todas aquellas universitarias de la Comunidad UNAM que tengan el deseo, la curiosidad y las ganas de aprender programación con el lenguaje Swift. Mediante el Club de Programadoras llamado: **"PWR GRL CODE".**

Misión:
Formar un grupo de universitarias con conocimientos sólidos en el lenguaje de programación Swift, de una manera divertida para ellas, con estrategias de aprendizaje dinámicas como lo serán las etapas del club de programación.
Igualmente, que se construyan en el camino agentes de cambio para la consecución de las futuras estudiantes de la universidad y de la facultad que den valor a la equidad, desarrollo, inclusión, cultura, diversidad y respeto.

Visión:
1.- Consolidarnos como el primer Club de Programadoras a nivel institucional.

2.- Hacer crecer el vínculo de la programación con el sector femenino de la facultad.

3.- Hacer equipos de trabajo capaces de participar en torneos de programación como Hackathones.

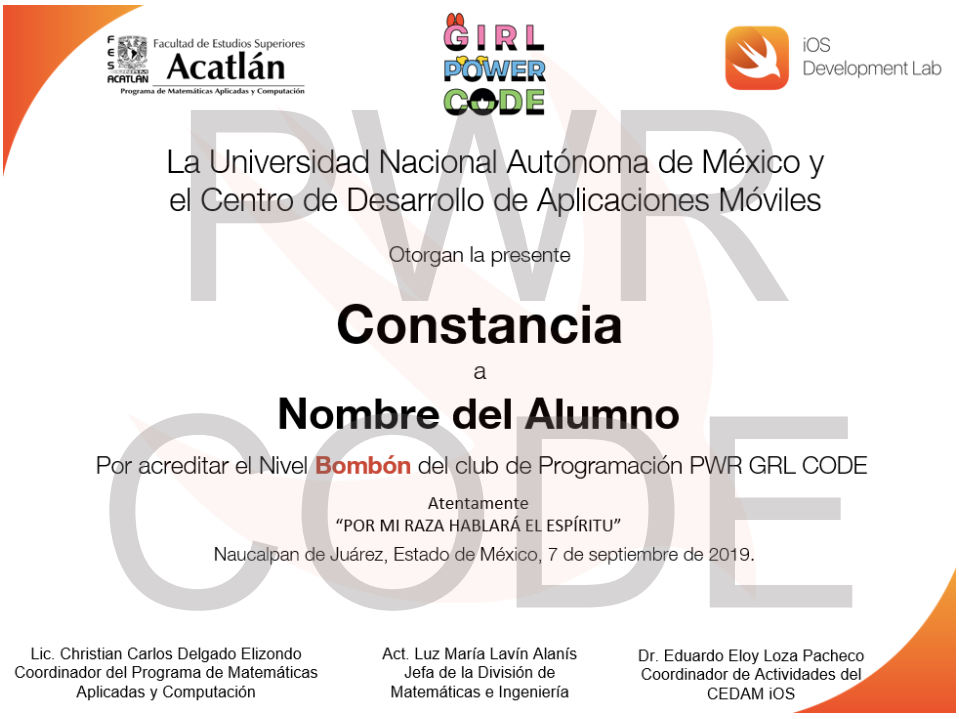
Descripción:
Dentro del club existen 3 niveles de aprendizaje: Burbuja, Bombón y Bellota. Para todos los niveles se es necesario haber tomado el curso de introducción al sistema operativo MAC OS-X que proporciona el *iOS Development Lab*. En la *Tabla 12* se muestra el contenido esquematizado de PWR CODE:

Tabla 12: Temario PWR GRL CODE 2019-2

Nivel	Descripción	Aprendizaje
Burbuja	Este nivel es para aquellas personas que comienzan su incursión en el lenguaje Swift. **Nota:** No se necesita pasar ninguna prueba para este nivel.	1.- Xcode. 2.- Variables y Constantes 3.- Tipos de datos. 4.- Declaraciones de datos. 5.- Manejo y manipulación de Strings. 6.- Manejo y manipulación de Números. 7.- Arrays 8.- If, Else, Else If, inline If 9.- For y While 10.- Manejo y manipulación de Números. 11.- Manejo de Booleanos.

Bombón	Para aquellas personas que tengan las bases básicas del lenguaje y comiencen con la etapa intermedia. **Nota:** Para conseguir la insignia de este nivel hay que pasar el test de prueba.	1.- Diccionarios. 2.- Matrices. 3.- Switch. 4.- Opcionales. 5.- Guard, If Guard. 6.- Funciones. 7.- Parámetros de las funciones.
Bellota	Este nivel es para los expertos de los tópicos de los dos niveles anteriores **Nota:** Para conseguir la insignia de este nivel hay que pasar el test de prueba.	1.- Objetos. 2.- Funciones con objetos. 3.- Propiedades. 4.- Manejo de propiedades. 5.- Herencia. 6.- ARKIT.

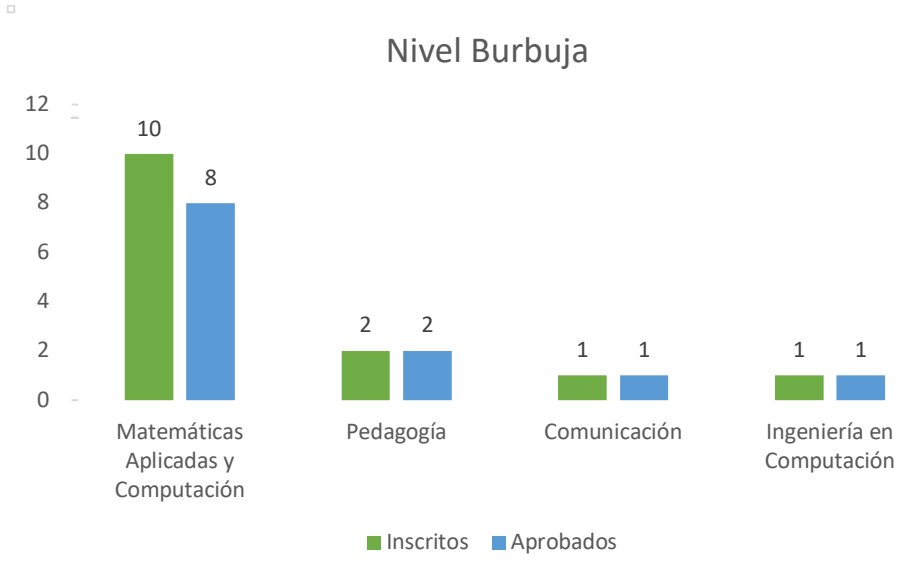
A continuación, en la *Imagen JJ* se muestra el diseño genérico de las constancias otorgadas a los estudiantes aprobados por nivel.



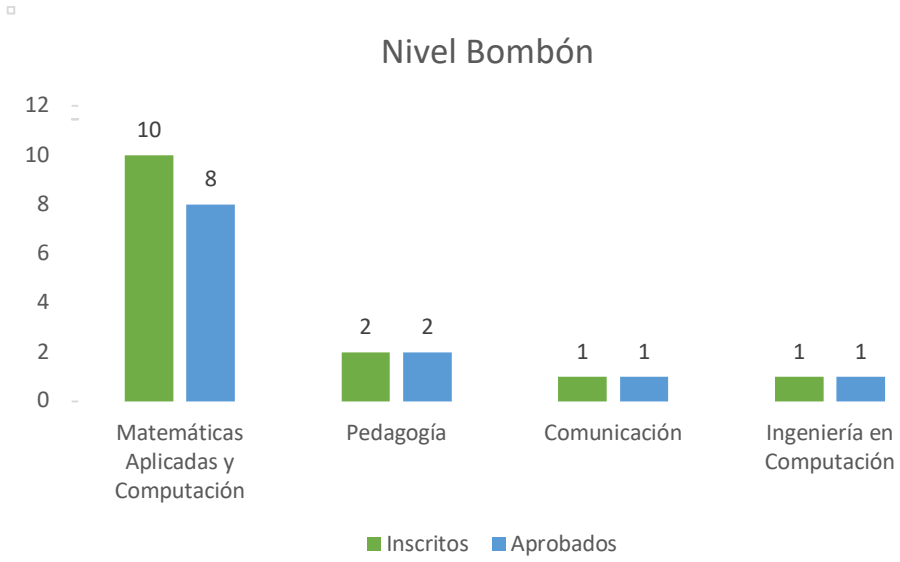
Contreras, A. (2019). Diseño de constancias de PWR GRL CODE 2019-2. [Imagen JJ]

A continuación, se observan los resultados del club de programación con respecto a sus evaluaciones finales durante la **Iteración 1** del proyecto, en las *Gráficas B y C*:

Gráfica B: Resultados nivel Burbuja de PWR GRL CODE



Gráfica C: Resultados nivel Bombón de PWR GRL CODE



Como se puede ver, más del 90% de los estudiantes inscritos fueron aprobados en sus respectivos niveles. El nivel Bellota en esta generación (2019-2) no tuvo alumnos inscritos debido a que el nivel de programación requerido en los estudiantes aún no se presentaba en la curva de aprendizaje del club de programación.

A continuación, se mostrarán una serie de fotografías del trabajo desempeñado por los mentores del *iOS Development Lab* dentro del club y los resultados de los estudiantes durante los periodos escolares 2019-2:

En la *Fotografía 6* se aprecia al primer grupo "piloto" del proyecto y a su mentora dando clases utilizando los equipos del *iOS Development Lab* en la F.E.S. Acatlán.

Fotografía 6: Mentoría en PWR GRL CODE

La *Fotografía 7* muestra a un grupo de estudiantes de diversas licenciaturas e instituciones UNAM, no sólo de la F.E.S. Acatlán que tomaron clases en el programa piloto de este proyecto.

Huerta, E. (2019). Clases en PWR GRL CODE. [Fotografía 7]

En la *Fotografía 8* se puede observar la entrega de constancias de la primera generación del club PWR GLR CODE, junto con su mentora y las autoridades correspondientes de la F.E.S. Acatlán.

Fotografía 7: Entrega de constancias de PWR GRL CODE

La *Fotografía 9* muestra a algunas estudiantes del programa piloto PWR GRL CODE siendo premiadas con mención honorífica por su destacable participación en el primer Hackathon *iOS Development Lab* 2019-2 y a su mentora. Lo que demostró que el club de programación ayudó al desarrollo tecnológico de las estudiantes de diferentes licenciaturas como Pedagogía y M.A.C. formando equipos de trabajo funcionales y multidisciplinarios con distintos enfoques gracias a los diversos pensamientos que se presentan con integrantes pertenecientes a diferentes carreras universitarias.

Fotografía 8: PWR GRL CODE en el Hackathon 2019-2

La *Fotografía 10* muestra la exposición del programa computacional desarrollado por el equipo “Bombón Swiftas” perteneciente a PWR GRL CODE, mientras es calificado por los jueces del Hackathon 2019-2.

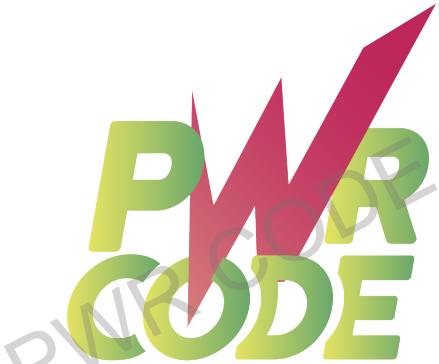
Fotografía 9: Exposición de PWR GRL CODE en el Hackathon 2019-2

Con el tiempo, el interés fue creciendo en toda la comunidad estudiantil, no sólo femenina, dentro de las distintas Licenciaturas e Ingenierías de la **F.E.S. ACATLÁN**. Teniendo un grupo grande de personas interesadas con distintos niveles de conocimientos en el área de la programación y SAP, se decidió realizar una retrospectiva de los resultados académicos obtenidos en la generación 2019-2. Se encontrando áreas de oportunidad en el proceso de enseñanza y dentro de la organización de trabajo del club PWR GRL CODE.

Posteriormente, se concluyó que se debía: 1) Mejorar los requerimientos mínimos planteados desde un inicio para lograr un mejor incremento potencial en el aprendizaje de los estudiantes como los temarios, 2) Establecer sistema de trabajo durante el proceso de desarrollo que permitiera detectar y organizar las tareas necesarias para llegar a los objetivos deseados, por lo que se decidió adoptar la metodología ***Scrum*** y 3) Crear una visión del club de programación para cubrir las nuevas necesidades presentadas por la comunidad UNAM. Para la generación 2020-1, el club de programación fue nombrado **PWR CODE**.

A continuación, se mostrará la Iteración 2, definida por: *Sprint* 0 y *Sprint* 1. En esta parte del proyecto se pueden observar las mejoras del club de programación.
Se mejoró la visión, misión, objetivos y el temario del club de programación adoptando las propuestas de la metodología *Scrum,* dando resultados académicos favorables. En la *Imagen KK* se muestra el logo del club de programación **PWR CODE**.

Club de Programación PWR CODE 2020-1



Contreras, A. (2019). Logo PWR CODE 2020-1. [Imagen KK]

Objetivos:
Integrar a todos aquellos universitarios de la Comunidad UNAM que tengan el deseo, la curiosidad y las ganas de aprender programación con el lenguaje Swift mediante el Club de Programadoras llamado: **"PWR CODE".**

Misión:
Formar un grupo de universitarios con conocimientos sólidos en el lenguaje de programación Swift, de una manera divertida para ellos, con estrategias de aprendizaje dinámicas como lo serán las etapas del club de programación.
Igualmente, que se construyan en el camino agentes de cambio para la consecución de las futuras estudiantes de la universidad y de la facultad que den valor a la equidad, desarrollo, inclusión, cultura, diversidad y respeto.

Visión:
1.- Consolidarnos como el mejor Club de Programación a nivel institucional.
2.- Hacer equipos de trabajo capaces de participar en torneos de programación como Hackathones.

Comentarios

Se puede observar que los elementos principales que guiaban al club de programación estaba pensado para lograr una inclusión en la comunidad universitaria, debido al gran interés de los alumnos por ser parte de ella.

Descripción:

Dentro del club llevaremos 4 niveles de aprendizaje: Burbuja, Bombón, Bellota y Sustancia X. En la *Tabla 13* se muestra el contenido esquematizado de PWR CODE:

Tabla 13: Temario PWR CODE 2020-1

Nivel	Descripción	Aprendizaje
Burbuja	Este nivel es para aquellas personas que comienzan su incursión en el lenguaje Swift. **Nota:** No se necesita pasar ninguna prueba para este nivel. Para conseguir la constancia de este nivel hay que pasar el test de prueba.	1.- ¿Qué es Swift? 2.- Introducción a Playgrounds. 3.- Tipos de datos. 4.- Declaraciones de variables. 5.- Manejo de operadores lógicos y aritméticos. 6.- Concatenación y sus diferentes formas. 7.- Colecciones: Array, Set, Diccionario. 8.- Condicionales: If, Else, Else If, inline If, Switch, Guard. 9.- Bucles: For, While y Repeat While.
Bombón	Para aquellas personas que tengan las bases básicas del lenguaje y comiencen con la etapa intermedia. **Nota:** Para conseguir la constancia de este nivel hay que pasar el test de prueba.	1.- Enum. 2.- Manejo de funciones. 3.- Clase, Herencias y Estructuras. 4.- Introducción a Aplicaciones Móviles con XCODE. 5.- Objetos: UIButton, Label, TextField, TextView, UIImage, Constrains, Scroll View. 6.- Navegation Controller, Segue, Paso por referencia. 7.- UISwitch.
Bellota	Este nivel es para los expertos de los tópicos de los dos niveles anteriores **Nota:** Para conseguir la constancia de este nivel hay que pasar el test de prueba.	1.- UISegment Control. 2.- UISlider. 3.- UIStipper. 4.- UIPicker View. 5.- Table View Controller 6.- Search Bar. 7.- Speaker. 8.- Manejo de Archivos TXT y Json.

Sustancia X	Este nivel es para los expertos de los tópicos de los tres niveles anteriores **Nota:** Para conseguir la constancia de este nivel hay que pasar el test de prueba.	1.- API, Framework, Librería, SDK. 2.- CocoaPods, Creación de Login, Firebase. 4.- Web Services. 5.- URL y Mapkit.

Comentarios

La estructura esquemática de los cursos cambió integrando un nuevo módulo. Además, con la retroalimentación obtenida en la generación "piloto", se ampliaron los contenidos dentro de los niveles Burbuja, Bombón y Bellota. En el caso de Burbuja se profundiza el nivel de conocimientos básicos en el lenguaje de programación Swift, así como en los Playgrounds. En Bombón se agregaron conceptos primordiales para dar inicio a la programación orientada a objetos y su uso dentro de XCODE usando la herramienta para la creación de aplicaciones móviles. Por otro lado, el nivel Bellota se concentró en la creación de aplicaciones móviles avanzadas, utilizando elementos cotidianos dentro de las aplicaciones móviles actuales. Finalmente, el módulo agregado se llamó Sustancia X en el cual tiene como objetivo ampliar los conocimientos sobre tópicos como lo que es un Framework, API, Librería, SDK, CocoaPods, Firebase, etcétera, dentro de XCODE.

A continuación, en la *Imagen LL* se mostrará el diseño genérico de las constancias que se les otorgaba a los estudiantes aprobados por nivel.

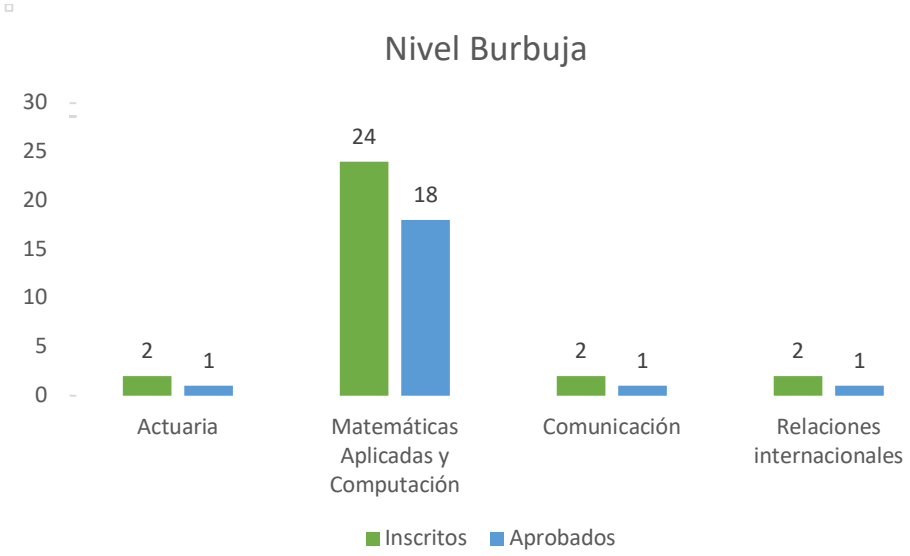
Contreras, A. (2019). Diseño de constancias de PWR CODE 2020-1. [Imagen LL]

Comentarios

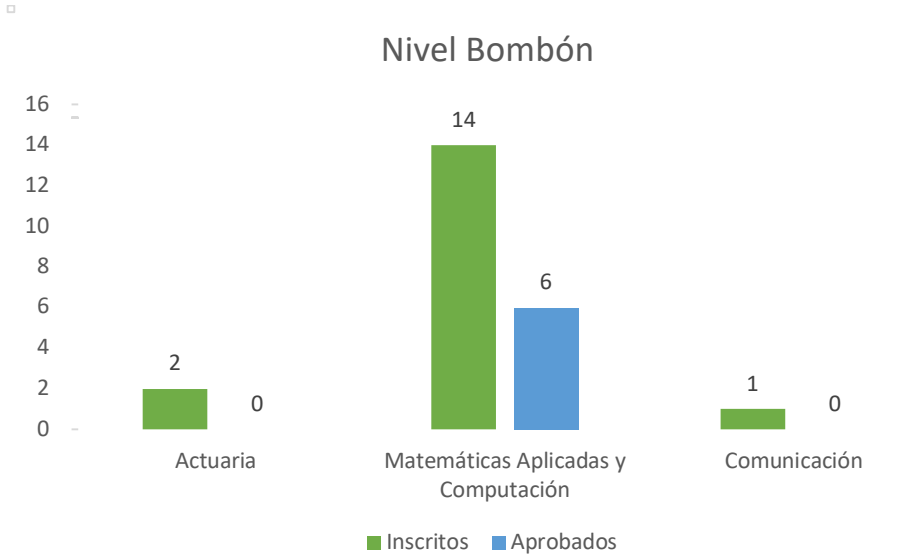
Se puede notar que hubo un incremento en los niveles y en sus tópicos. Cambios necesarios para elevar los niveles de programación de los alumnos y una nueva visión, misión y objetivos que atendiera las necesidades de toda la comunidad UNAM dentro de la F.E.S. Acatlán.

A continuación, se mostrarán los resultados del club de programación con respecto a sus evaluaciones finales durante la Iteración 2 del proyecto, en las *Gráficas D, E, F y G*:

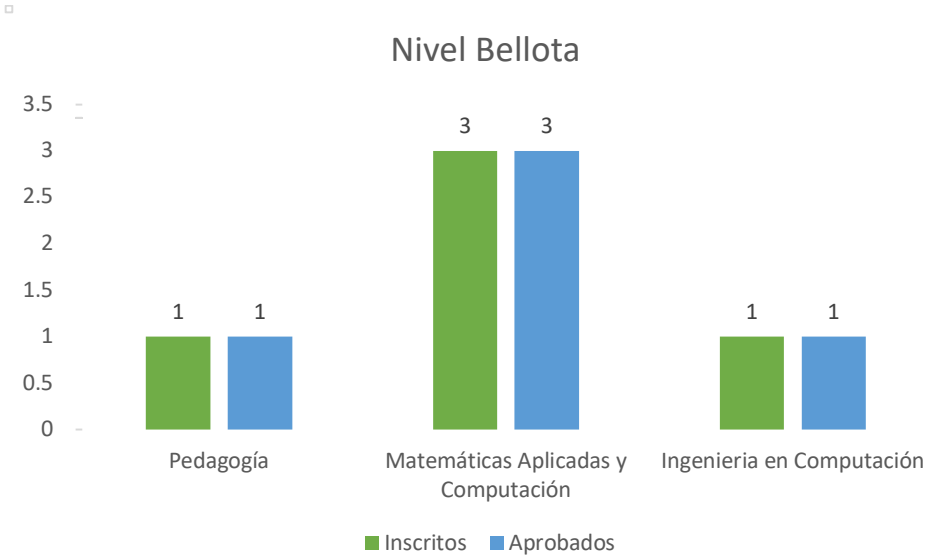
Gráfica D: Resultados nivel Burbuja de PWR CODE



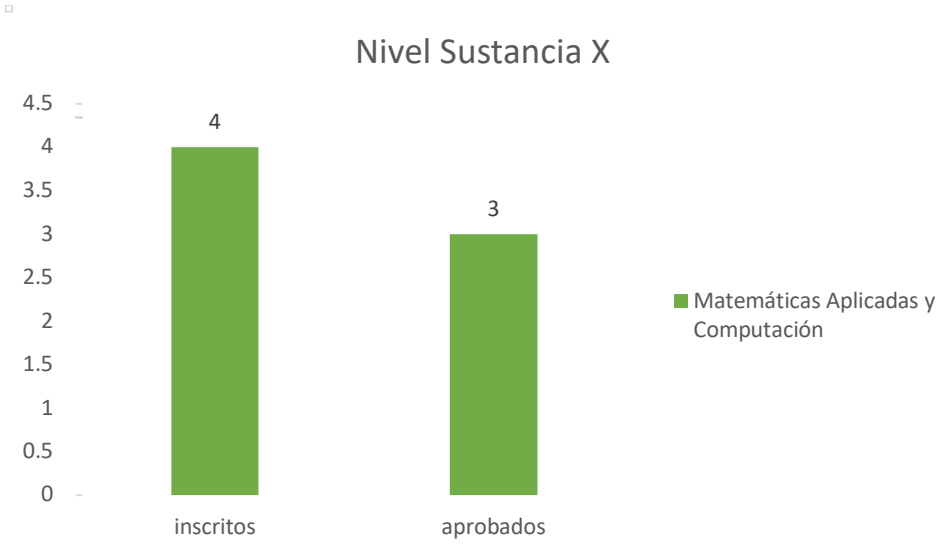
Gráfica E: Resultados nivel Bombón de PWR CODE



Gráfica F: Resultados nivel Bellota de PWR CODE



Gráfica G: Resultados nivel Sustancia X de PWR CODE



Como se puede observar, en algunos casos, mínimo el 50% de los estudiantes inscritos fueron aprobados en sus respectivos niveles. El nivel Bellota y Sustancia X en esta generación (2020-2) tuvieron alumnos inscritos debido a que el nivel de programación en los estudiantes ya era más avanzado por el tiempo de madures del club de programación. Esto nos permitió observar las áreas de oportunidad en donde podíamos mejorar en la próxima iteración.

A continuación, se mostrarán una serie de fotografías del trabajo desempeñado por los mentores del *iOS Development Lab* dentro del club y los resultados de los estudiantes durante los periodos escolares 2020-1:

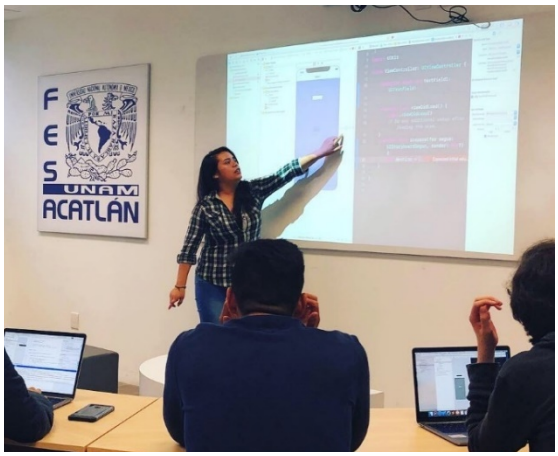
En la *Fotografía 11* se observa a los nuevos estudiantes del ciclo escolar 2020-1 del nivel Burbuja apoyándose del material ***"Mis primeros pasos en Swift".***

Fotografía 10: PWR CODE trabajando con la guía de aprendizaje

Para el ciclo escolar 2020-1, PWR CODE creció y los mentores encargados de cada nivel y grupo dentro del club cambiaron. En la *Fotografía 12 y 13* se muestra a las nuevas mentoras impartiendo nivel Bombón. Para dar este nivel las mentoras fueron preparadas a un entrenamiento previo por la encargada anterior de PWR GRL CODE, en donde se explicaba la dinámica de las clases, el temario, la visión y objetivos del club.

Fotografía 12: Nivel Bombón PWR CODE grupo "Peludito"

Fotografía 11: Nivel Bombón PWR CODE grupo "Él"



Un efecto colateral del ciclo escolar 2019-2 fue que las estudiantes graduadas del club de programación se postularan como mentoras para dar clases en PWR CODE, algunas pertenecían al servicio social del *iOS Development Lab* y algunas otras simplemente por diversión y pasión por compartir sus conocimientos. En la *Fotografía 14 y 15* se muestran a las estudiantes ya mencionadas.

Fotografía 14: Curso de introducción a Swift

Fotografía 13: Nivel Burbuja PWR CODE grupo "Banda Gangrena"

En la *Fotografía 16* también se aprecia a los estudiantes nivel burbuja generación 2020-1 y a los mentores a los costados.

Fotografía 15: Entrega de constancias nivel Burbuja en PWR CODE

En la *Fotografía 17* se observa a un grupo nivel Burbuja aplicando el examen final del nivel para conseguir su constancia.

Fotografía 16: Aplicación de exámenes en PWR CODE

En las siguientes *Fotografías 18, 19 y 20* se muestra el cierre de curso de PWR CODE 2020-1 de los diferentes niveles existentes y a los estudiantes con sus constancias.

Fotografía 17: Nivel Burbuja grupo "Banda Ameba"

Fotografía 18: Nivel Burbuja grupo "Banda Gangrena"

Fotografía 19: Nivel Bombón grupo "Él"

En las siguientes *Fotografías 21* se observa a los estudiantes y académicos cantando al unísono el “Goya” en el cierre de curso de PWR CODE 2020-1.

Fotografía 20: Cierre de curso de PWR CODE 2020-1

¡GOYA! ¡GOYA!

¡CACHUN, CACHUN, RA, RA!

¡CACHUN, CACHUN, RA, RA!

¡GOYA!

¡¡UNIVERSIDAD!![6]

[6] Fundación UNAM. (2020). ¡Goya! La historia del grito universitario. UNAM Sitio web: https://www.fundacionunam.org.mx/cancha-puma/goya-la-historia-del-grito-universitario/

7.2 Nuevas propuestas

En el capítulo VI se muestra paso a paso la propuesta de aplicación del club de programación para las generaciones futuras. De igual manera, se mostró que, al desarrollar la visión del proyecto, que es uno de los procesos de Scrum durante la fase de inicio, surgieron las siguientes propuestas. Destacando que, se requirió de un trabajo conjunto y una reunión con los interesados en el proyecto para la definición de la visión, misión, objetivos y temario que siguió el club de programación PWR CODE. A continuación, se mostrará el documento resultante del proceso de creación de la visión del proyecto para lo que será una futura iteración 3. El logo del club de programación que fue diseñado en enero 2020 y se muestra en la Imagen Z:

Club de programación PWR CODE

Rangel D. (2020). Logo de PWR CODE. [Imagen Z]

Objetivos:
Despertar el interés de las ramas tecnológicas en alumnos de todas las áreas de estudio dentro de la F.E.S. Acatlán, a través del proceso de enseñanza de la **SAP** aplicado a la programación, por medio del conjunto de herramientas tecnológicas disponibles del Sistema Operativo macOS como lo es Xcode, con el uso del lenguaje de programación Swift. Se plantea lo anterior con el propósito de crear soluciones (aplicaciones móviles) a problemáticas dentro de la F.E.S. Acatlán y la sociedad, a través de los conocimientos obtenidos en PWR CODE.

Misión:

Formar universitarios de todas las áreas de estudio dentro de la F.E.S. Acatlán con conocimientos sólidos en el lenguaje de programación Swift de una manera **divertida**, fuera de las aulas académicas. Con estrategias de aprendizaje didácticas y a lo largo de cada uno de los niveles del club de programación, se formarán

alumnos competitivos en la rama de las ciencias de la computación mediante cada uno de los niveles del club de programación. De igual manera, se pretende construir en el camino agentes de cambio para las futuras generaciones dentro de la Facultad que den valor a la **equidad, desarrollo, inclusión, cultura, diversidad y respeto**.

Visión:
1.- Consolidarnos como el mejor club de programación a nivel UNAM, mediante propuestas didácticas de aprendizaje actualizadas a la par de las tecnologías.

2.- Hacer crecer el vínculo de la programación y la SAP con las diversas áreas dentro de la F.E.S. Acatlán, en especial, con la licenciatura de M.A.C.

3.- Hacer equipos de trabajo capaces de participar en torneos de programación como Hackathones.

4.- Crear interés por resolver problemáticas dentro de la F.E.S. Acatlán a través de los conocimientos obtenidos en PWR CODE.

Temario:
Dentro del club se llevan 3 niveles de aprendizaje: Burbuja, Bombón y Bellota.
Para todos los niveles es necesario haber tomado el curso de introducción al sistema operativo macOS y otros requisitos dependiendo del nivel. En la *Tabla 15* se muestra el contenido esquematizado de PWR CODE:

Tabla 14: Temario PWR CODE 2020-2

Nivel	Descripción	Aprendizaje
[7] 	Este nivel es para aquellas personas que comienzan su incursión en el lenguaje Swift. **Requisitos:** No se necesita pasar ninguna prueba para este nivel. **Insignia Nivel Burbuja:** Únicamente podrán obtenerla los estudiantes que aprueben el nivel con calificación aprobatoria en el examen final.	1.- Todos los temas vistos en la guía de aprendizaje “Mis primeros pasos en Swift”. 2.- Introducción a Git, GitHub. 3.- SAP. 4.-Trabajo en Equipo con repositorio remoto. (GitHub)

[7] Rangel, D. (2020). Logos de PWR CODE 2020-2.

8 	El siguiente nivel es para aquellas personas que tengan las bases básicas del lenguaje de programación Swift y comiencen con la etapa intermedia. **Requisitos:** Haber obtenido la *Insignia Nivel Burbuja* o haber aprobado un examen de conocimientos en el lenguaje de programación y la solución algorítmica de problemas. **Insignia Nivel Bombón:** Únicamente podrán obtenerla los estudiantes que entreguen el proyecto final propuesto por sus mentores y este obtenga una calificación aprobatoria.	1.- Introducción a Apps, SwiftUI. 2.- Clase, Herencia y Estructuras. 3.- Objetos: UIButton, Label, TextField, TextView, UIImage, Constrains, UISwitch, UISegmente Control, UISlider, UIPicker View. 4.- Navegation Controller, Segue, Paso por referencia. 5.- Audio y Animaciones 6.- ¿Cómo subir mi aplicación a la App Store?
9 	Este nivel es para los expertos de los tópicos de los dos niveles anteriores **Requisitos:** Haber obtenido la *Insignia Nivel Bombón* o haber aprobado un examen de conocimientos en el lenguaje de programación orientado a objetos y la solución algorítmica de problemas. **Insignia Nivel Bellota:** Únicamente podrán obtenerla los estudiantes que entreguen el proyecto final propuesto por sus mentores y este obtenga una calificación aprobatoria.	1.- Tipos de Páginas: Tab Bar, Page Control, UIScroll View. 2.- API, Framework, Librería, SDK. 3.- CocoaPods. 4.- Bases de Datos (Firebase). 5.- URL. 6.- Introducción a ARKIT.

[8-9] Rangel, D. (2020). Logos de PWR CODE 2020-2.

Comentarios

El siguiente esquema de trabajo se muestran las siguientes modificaciones. Para el nivel Burbuja se realizó la integración de la guía de aprendizaje “Mis primeros pasos en Swift” contempla los conceptos esenciales que todo desarrollador debe dominar como las bases del lenguaje de programación Swift y la SAP que busca sistematizar la creación de algoritmos. También se puede observar que con el propósito de tener un trabajo colaborativo entre los estudiantes se agregó la tecnología de cómputo Git. Git es un manejador de versiones esencial en los equipos de trabajo de software. Para el nivel Bombón se orientó al desarrollo de aplicaciones móviles, introducción a SwiftUI. SwiftUI consiste en reducir el número de líneas de código mediante la filosofía WYSIWYG. [24] Finalmente, para el nivel Bellota se agregó la línea de conocimiento de ARKit, que es un conjunto de componentes que permite la graficación por computadora usando el novedoso concepto de realidad aumentada en las aplicaciones móviles o videojuegos. [25]

Finalmente, se puede concluir lo siguiente. El club imparte desde temas básicos hasta avanzados, aprovechando todos los recursos tecnológicos que ofrece el Centro de Desarrollo de Aplicaciones Móviles – *iOS Development Lab* que hay en la institución. Con el objetivo de que los estudiantes desarrollen aplicaciones móviles. Cabe destacar que el temario del Nivel Burbuja en su mayoría se encuentra en la guía de aprendizaje: “Mis primeros pasos en Swift” previamente descrita aquí, en donde el mentor podrá actualizar los temas sí se llegase a tener una actualización nueva en el lenguaje de programación. Para los niveles posteriores se da una propuesta de temas, igualmente el mentor es libre de agregar o modificar el orden de enseñanza de algún tópico. A continuación, se muestra en la *Imagen AA* la insignia digital de cada nivel de programación que se les otorgará a los alumnos que logren aprobar los cursos.

Rangel D. (2020). Insignias PWR CODE 2020. [Imagen AA]

En la *Imagen MM* se muestra el diseño de la hoja de registro de actividades y tareas, en donde serán marcadas con un sello si cumplen con los requisitos que indicó el mentor. Está forma de evaluación es una propuesta opcional para cada mentor de cada nivel dentro de PWR CODE.

Imagen MM: Hoja de registro de actividades y tareas nivel Burbuja

Conclusiones

El objetivo fundamental de este proyecto de tesis fue resolver la necesidad universitaria de utilizar nuevas e innovadoras herramientas tecnológicas de aprendizaje que Apple proporciona a la F.E.S. Acatlán con la ayuda de los recursos tecnológicos del *iOS Development Lab*, prevaleciendo los recursos y visión de los *iOS Development Lab* y al mismo tiempo incrementando el desarrollo institucional UNAM (Plan 2019-2023) [11] mediante el desarrollo de un proyecto educativo. Este proyecto consistió en la creación de un club de programación llamado PWR CODE. Para lograr un proceso óptimo y estructurado se decidió que fuera aplicado a una de las metodologías ágiles más importantes, *Scrum*, cuyo interés fue medir e incrementar la efectividad en el uso de los recursos académicos, científicos y tecnológicos. En PWR CODE se utilizaron herramientas como lo son los Playgrounds dentro del IDE de XCODE, en donde se empleó el lenguaje de programación Swift para la enseñanza y reforzamiento de las bases teóricas de la programación mediante la solución algorítmica de problemas. La SAP, es parte fundamental de la programación en las Ciencias de la Computación y se encuentra en una materia curricular impartida en las aulas del primer semestre de la Licenciatura en M.A.C. en la Facultad.

Además, este proyecto de tesis tuvo la finalidad de expandir la experiencia del club de programación a más estudiantes, y así poder aprender y mejorar en cada iteración para elevar los estándares de aprendizaje, lograr mantener la calidad para futuras generaciones con un modelo de trabajo flexible que se mantenga siempre a la vanguardia para identificar, promover, desarrollar, reforzar el liderazgo y habilidades técnicas sobresalientes. Tal como sucede en los centros de liderazgo de Apple [2] y la UNAM donde se usan como **pilares esenciales el aprendizaje, la enseñanza y el entorno.**

Durante los inicios del club de programación se observó a jóvenes de distintas Licenciaturas e Ingenierías de la F.E.S. Acatlán y de otras Facultades dentro de la UNAM, interesados en el aprendizaje del lenguaje de programación Swift. Estos jóvenes tuvieron acceso a un conocimiento básico, intermedio y avanzado, dando como resultado una formación amplia y completa que, con el tiempo, les permitió desarrollar aplicaciones móviles y ser participantes de concursos de programación avanzada como los Hackathones organizado cada semestre por las autoridades de la coordinación de M.A.C. y el *iOS Development Lab*. De igual manera, en PWR CODE se tuvo como mentores a personas altamente capacitadas y con experiencia en el lenguaje de programación Swift, con la finalidad de hacer crecer las habilidades tecnológicas de los estudiantes y reforzar sus áreas de oportunidad por medio de un **ambiente moderno, cómodo y divertido** fuera de las aulas.

Por consiguiente, la aportación principal del trabajo de tesis consistió en el diseño, desarrollo e implementación de un programa sistemático para un club de programación llamado PWR CODE. Sin embargo, una vez definidos los procesos de creación de este club, es importante resaltar que, se pueden ampliar para otro tipo de clubes en el marco científico y tecnológico que privilegien el aprendizaje con el uso de herramientas similares a las mencionadas en los capítulos 2, 3 y 4 de este trabajo de tesis. De la misma forma, el manejo de identidad, propósitos y objetivos es necesaria para la puesta en marcha de un proyecto de esta naturaleza, por lo cual se aportó la visión y misión de PWR CODE. Asimismo, se trabajó de manera colaborativa con el *iOS Development Lab* Manager en un esquema de trabajo que se caracterizó en un temario. Del temario se desarrollaron los tópicos necesarios para desarrollar las habilidades esperadas en los estudiantes al término del curso y por consecuencia se elaboró una guía de aprendizaje llamada **"Mis primeros pasos en Swift"**, elaborada en la herramienta de Playgrounds, para el nivel básico del club de programación. Este material didáctico fue utilizado para realizar las clases más llamativas e interesantes para los estudiantes, agregando contenido multimedia que tuviera sentido con el temario de programación.

PWR CODE se convirtió en el primer club de programación de la **F.E.S. Acatlán** que creo soluciones a problemáticas dentro de la Facultad desarrollando aplicaciones móviles en **el lenguaje de programación Swift** a través de la **SAP** por medio del conjunto de herramientas tecnológicas disponibles del Sistema Operativo macOS como lo es **Xcode**. Abierto a toda la comunidad estudiantil apoyándose de los recursos tecnológicos del *iOS Development Lab* dando como beneficios:

- El aprovechamiento de los recursos tecnológicos que ofrecen los ***iOS Development Labs***.
- Despertar el interés de las ramas tecnológicas a alumnos de todas las áreas de estudio dentro de la F.E.S. Acatlán.
- Finalizar los cursos de PWR CODE con alumnos competitivos en las ramas tecnológicas capaces de participar en torneos de programación como Hackathones.
- Optimizar los tiempos de aprendizaje y ampliar la experiencia educativa a más estudiantes con el apoyo de una organización sólida, a través de la metodología ágil Scrum.
- Elevar y actualizar constantemente los estándares de aprendizaje mediante cada nueva iteración dentro del Sprint de implementación de PWR CODE.
- Mantener un modelo de trabajo para generaciones futuras mediante este proyecto de tesis.

- Crear interés entre los estudiantes de PWR CODE por resolver problemáticas dentro de la F.E.S. Acatlán a través de los conocimientos obtenidos en el club de programación.

Finalmente podemos afirmar que, es posible mejorar la calidad y tiempos de aprendizaje en la enseñanza a la programación, a través de formas didácticas como lo son hoy en día las herramientas tecnológicas que nos ofrecen los entornos de desarrollo de Apple, utilizando una de las metodologías ágiles como lo es Scrum para la planeación, desarrollo, mantenimiento y mejora iterativa de programas escolares como lo es un club de programación.

Glosario

A

Agilidad
Dentro del campo de las metodologías ágiles, el concepto de agilidad hace referencia a un comportamiento persistente o habilidad que presenta flexibilidad para adaptarse a los cambios esperados o inesperados, rápidamente; persigue la duración más corta en tiempo, usa instrumentos económicos 17, 21

Agreed Actionable Improvements
Mejoras accionables aceptadas que se acuerdan en la reunión de Sprint Retrospective 34, 82

Algoritmo
Serie de pasos, procedimientos o acciones que permiten alcanzar un resultado para resolver un problema o llegar a una meta . 37, 38, 40, 42, 43, 44, 46, 47, 105

Aplicación móvil
Ees una aplicación informática diseñada para ser ejecutada en teléfonos inteligentes, tabletas y otros dispositivos móviles. 50, 73

Artefactos
Los artefactos del marco de scrum técnico son Product backlog, sprint backlog, sprint y el resultado de cada sprint. 8, 17

B

Backlog Refinement
Como parte del flujo continuo de análisis de las historias de usuario, scrum se complementa con esta actividad de mantenimiento de la pila de producto para mantenerla actualizada .. 4, 29, 81

Blockers
Obstáculos que se pueden llegar a presentar durante el proyecto 34, 80

Burn down
es el gráfico que actualiza el equipo en las reuniones de seguimiento del sprint, para monitorizar el ritmo de avance, y detectar de forma temprana posibles desviaciones sobre la previsión que pudieran comprometer la entrega al final de sprint 33, 34

D

Daily Standup
Reunión diaria del Equipo de Scrum para dar actulización de su trabajo o presentar si tienen problemas para continuar ... 4, 29, 34, 36, 63, 78, 80

Deuda técnica
Es aquella que se adquiere al producir código pobre, incumpliendo prácticas aconsejadas para el desarrollo de software 69

Diagramas de Flujo
Representación gráfica de un algoritmo o proceso, y que se realizan bajo símbolos 43, 59

E

Empirismo
Doctrina psicológica y epistemológica que, frente al racionalismo, afirma que cualquier tipo de conocimiento procede únicamente de la experiencia, ya sea experiencia interna (reflexión) o externa (sensación), y que esta es su única base 23

Épica
Se denomina Epic a una historia de usuario que por su gran tamaño, el equipo descompone en historias con un tamaño más adecuado para ser gestionada con los principios y técnicas ágiles estimación y seguimiento cercano (normalmente diario) 29

Estimación de tareas
Es una práctica ágil, para conducir las reuniones en las que se estima el esfuerzo y la duración de tareas.. 3, 4, 32, 33, 36, 63, 70, 72, 73, 75, 77, 80, 81

H

Herramientas cognitivas
Las herramientas cognitivas son aquellas que permiten aprender de manera esquematizada, que apoyan, guían y extienden los procesos de pensamiento de los alumnos 12, 42, 43

Historias de usuario
Descripción de una funcionalidad que debe incorporar un sistema de software, y cuya implementación aporta valor al cliente 3, 29, 31, 32, 68, 70, 72, 74, 75, 77, 78

I

IDE
Es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica 48

M

P

R

S

X

Referencias

[1] Apple Education. (2019). Apple Distinguished Schools. De Apple Inc. Sitio web: https://www.apple.com/education/apple-distinguished-schools/

[2] Apple Education. (2018). Diseñar el Aprendizaje, la Enseñanza y el Entorno Educativo con Apple. En Innovación en las Escuelas (41). Global: Apple Inc.

[3] Backhouse, R. (2011). Algorithmic Problem Solving. U.K: The University of Nottingham.

[4] Blokehead, T. (2016). ¡Guía definitiva de Scrum! E.U.A: Babelcube Inc.

[5] Apple Education. (2018). Introducción al desarrollo de aplicaciones con Swift. E.U.A: Apple Inc.

[6] Trigas, M. (2018). Gestión de Proyectos Informáticos. México.

[7] SCRUMstudy. (2016). SMC Libro de Ejercicios Alumno. E.U.A: VMEdu, Inc. V1.0.

[8] Cunningham, W. (2001). Manifesto for Agile Software Development. 2019. De Ward Cunningham Sitio web: https://agilemanifesto.org

[9] Universidad Nacional Autónoma de Ciudad Juárez. (2018). Metodologías. 2019. De Universidad Nacional Autónoma de Ciudad Juárez Sitio web: http://www.uacj.mx/CGTI/CDTE/JPM/Documents/IIT/proceprogra/metodologias.html

[10] Chamas, F. (6 febrero 2019). iOS Development Lab: una puerta a la programación. UNIVERSIDAD PANAMERICANA Sitio web: https://medialab.up.edu.mx/noticias/ios-development-lab-una-puerta-a-la-programacion/

[11] Graue, L. (Ciudad Universitaria, octubre de 2019). Síntesis del proyecto de trabajo. UNAM Sitio web: http://www.juntadegobierno.unam.mx/rector2019/files/DR-ENRIQUE-GRAUE/EGW_Sintesis_proyecto.pdf

[12] Singh, Y and Malhotra, R. (2012). Object-Oriented Software Engineering. New Dheli, India: PHI Learning.

[13] Anónimo. (2020). Símbolos de diagramas de flujo. Smartdraw Sitio web: https://www.smartdraw.com/flowchart/simbolos-de-diagramas-de-flujo.htm

[14] Anónimo. (2019). Algoritmo en Informática. Concepto.de Sitio web: https://concepto.de/algoritmo-en-informatica/

[15] Real Academia Española. (2019). Diccionario de la Lengua Española. de Real Academia Española Sitio web: https://dle.rae.es/seudo-

[16] Grossman, P. (2002). Discrete Mathematics for Computing. New York: PALGRAVE MACMILLAN.

[17] Levitin, A. (2012). Introduction to the Design and Analysis of Algorithms. Villanova University: Pearson.

[18] Soporte Apple Developer. (2020). Xcode. Apple Inc. Sitio web: https://developer.apple.com/es/support/xcode/

[19] Apple Education. “Introducción al desarrollo de apps con Swift”. Apple Inc. - Education, 2017. Apple Books. Sitio web: https://books.apple.com/mx/book/introducci%C3%B3n-al-desarrollo-de-apps-con-swift/id1216831475

[20] Apple Developer. (2020). Markup Formatting Reference. 22/03/2020, de Apple Inc. Sitio web: https://developer.apple.com/library/archive/documentation/Xcode/Reference/xcode_markup_formatting_ref/index.html#//apple_ref/doc/uid/TP40016497-CH2-SW1

[21] Apple Developer. (2020). Swift - Apple (MX). Apple Inc. Sitio web: https://www.apple.com/mx/swift/

[22] Apple Developer. (2020). Welcome to Swift.org. Apple Inc. Sitio web: https://swift.org

[23] Apple Developer. (2020). Xcode 11 Release Notes. Apple Inc. Sitio web:https://developer.apple.com/documentation/xcode_release_notes/xcode_11_release_notes

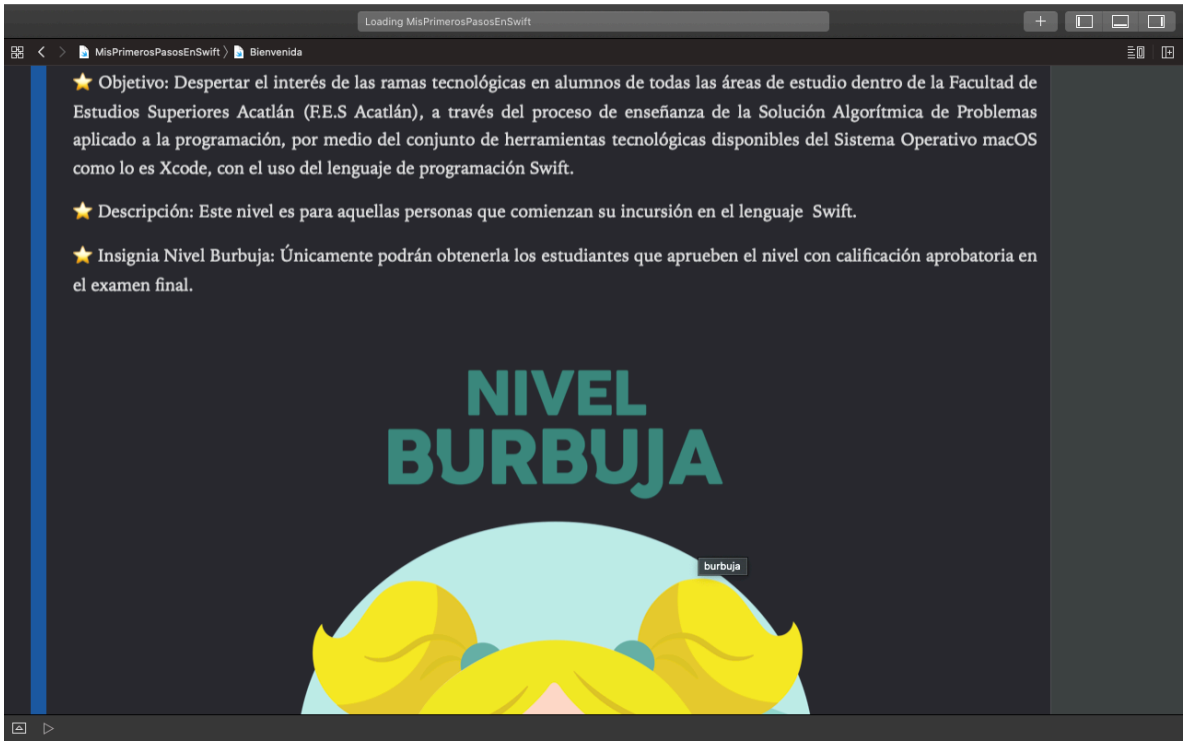
[24] Tam, A. (2019). SwiftUI: Getting Started. raywenderlich.com Sitio web: https://www.raywenderlich.com/3715234-swiftui-getting-started

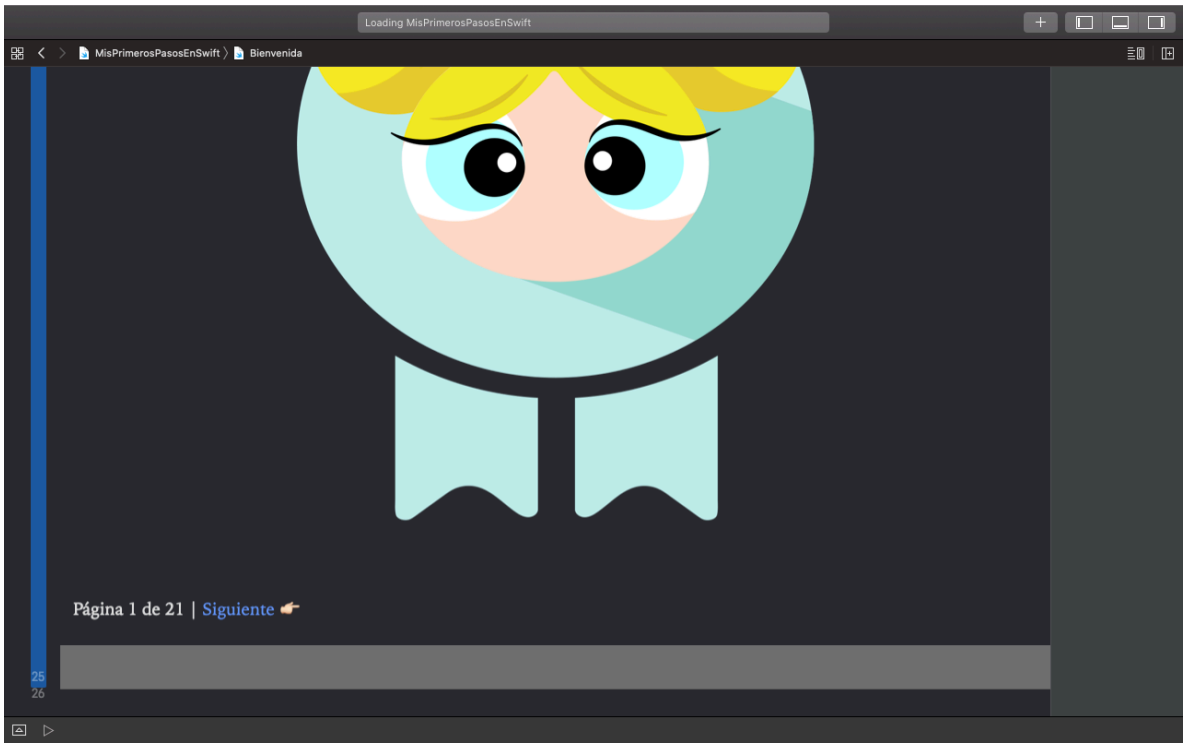
[25] Apple Developer. (2020). ARKit. Apple Inc. Sitio web: https://developer.apple.com/documentation/arkit

Anexos


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Bienvenida
¡Bienvenidos!
PWR
CODE

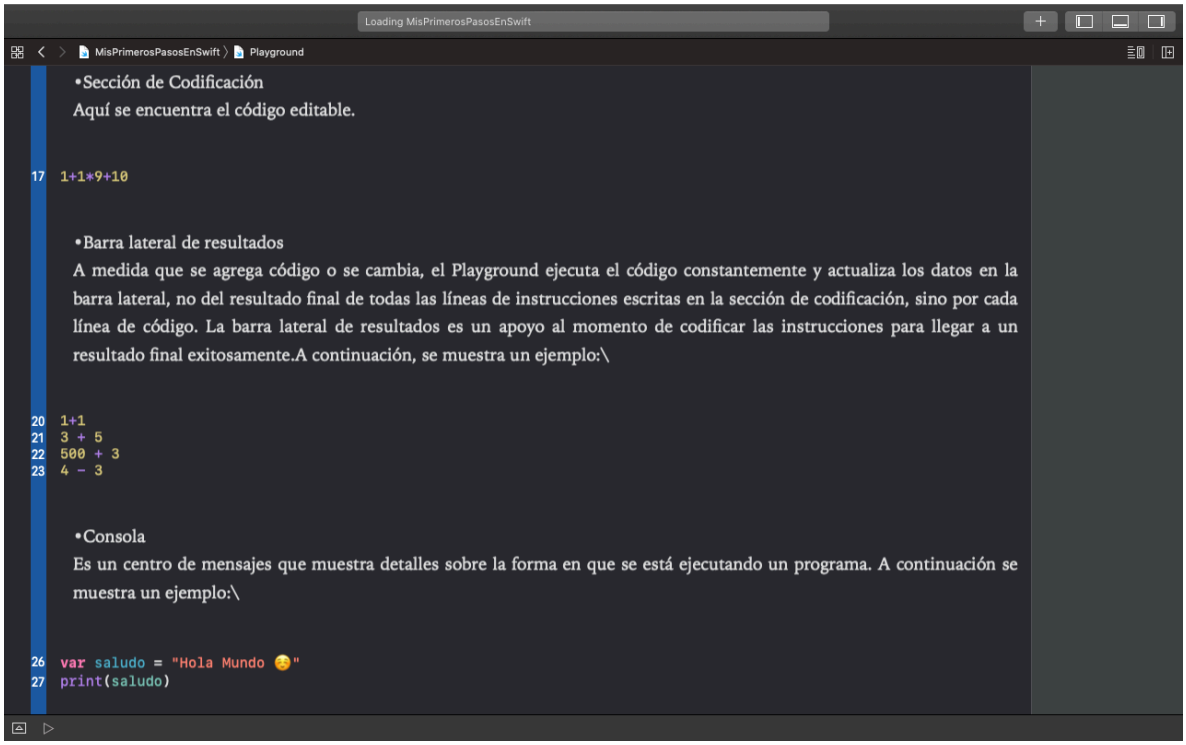

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Bienvenida
Nivel Burbuja
burbuja
NIVEL
BURBUJA

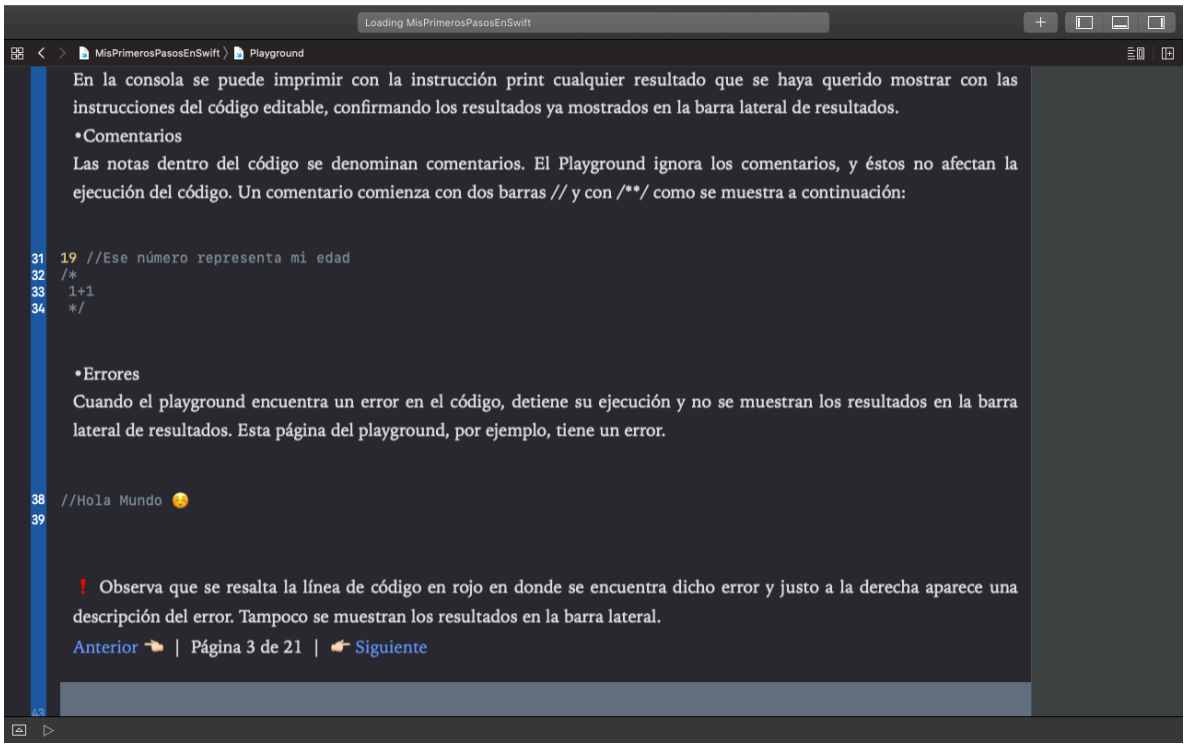

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Bienvenida
⭐ Objetivo: Despertar el interés de las ramas tecnológicas en alumnos de todas las áreas de estudio dentro de la Facultad de Estudios Superiores Acatlán (F.E.S Acatlán), a través del proceso de enseñanza de la Solución Algorítmica de Problemas aplicado a la programación, por medio del conjunto de herramientas tecnológicas disponibles del Sistema Operativo macOS como lo es Xcode, con el uso del lenguaje de programación Swift.
⭐ Descripción: Este nivel es para aquellas personas que comienzan su incursión en el lenguaje Swift.
⭐ Insignia Nivel Burbuja: Únicamente podrán obtenerla los estudiantes que aprueben el nivel con calificación aprobatoria en el examen final.
NIVEL BURBUJA
burbuja

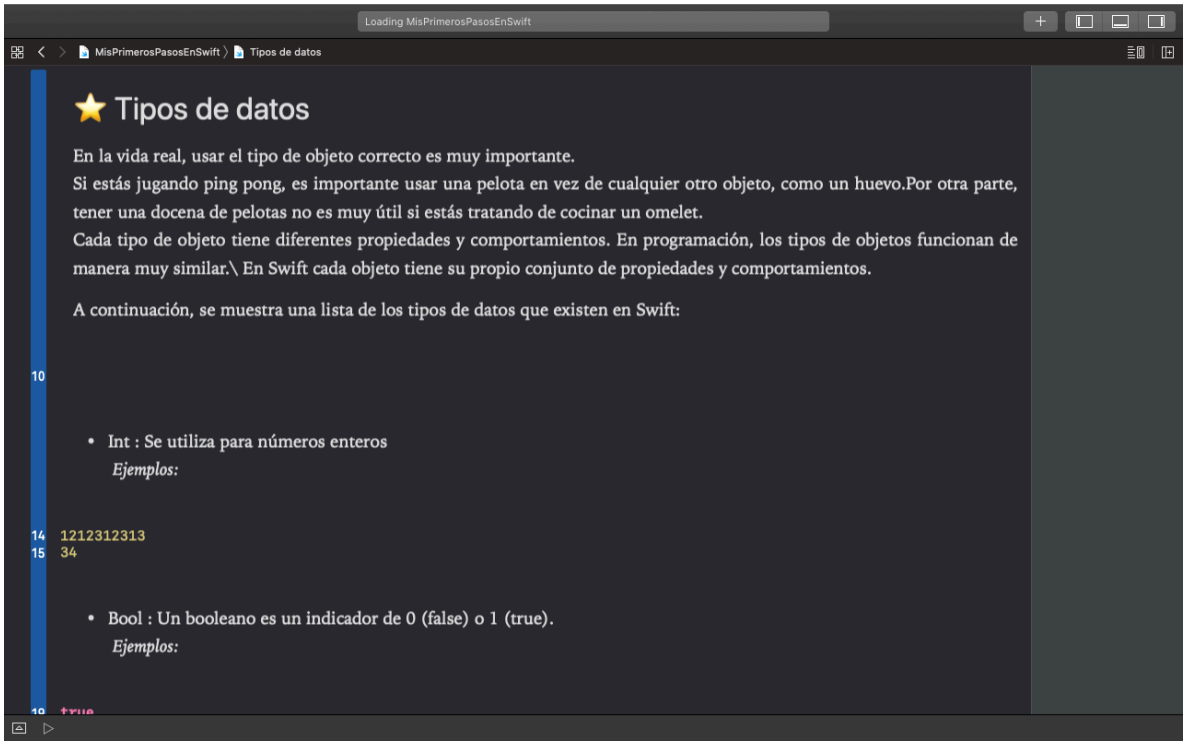

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Bienvenida
Página 1 de 21 | Siguiente

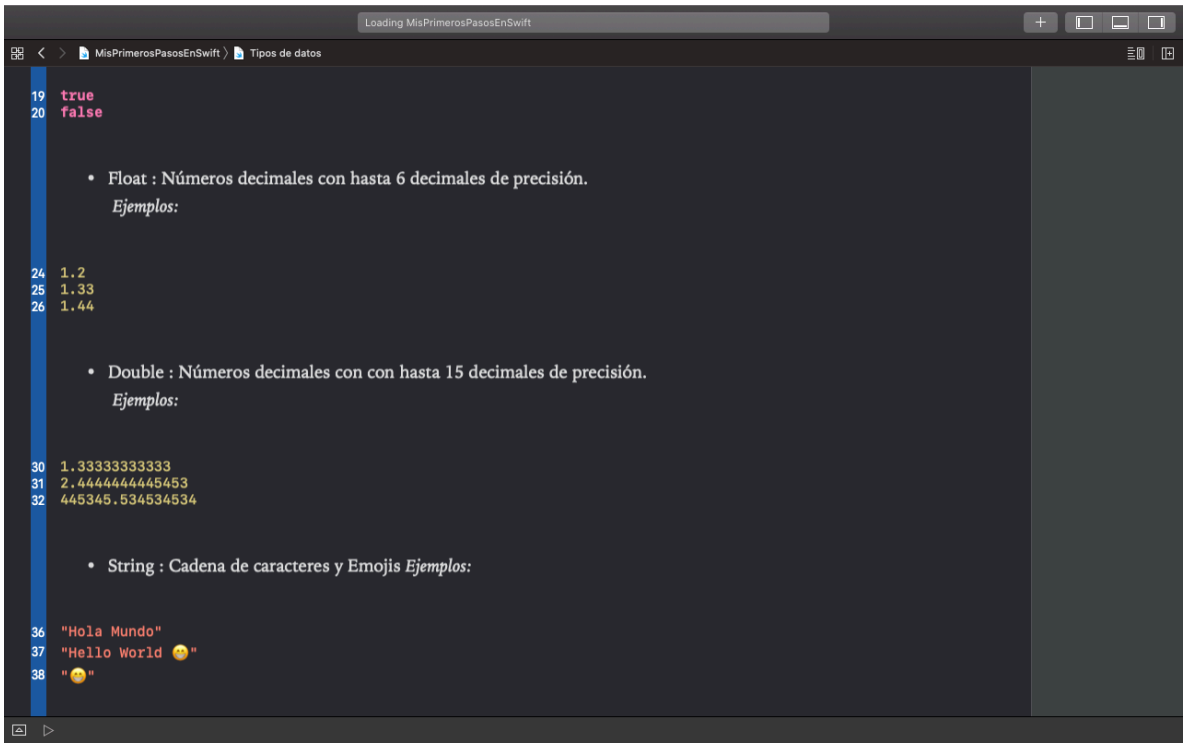

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Evaluación
EVALUACIÓN:
100% Examen Final (Necesario acreditarlo con 7)
PUNTOS EXTRAS:
Actividades -> 2 décimas cada uno (máximo 5)
Tareas -> 5 decimas cada una
A continuación, se muestra la hoja en donde se llevará el registro de tus actividades y tareas, estas serán marcadas con un sello si cumplen con los requisitos que indico tu mentor.
Nombre:
Anterior | Página 2 de 21 | Siguiente

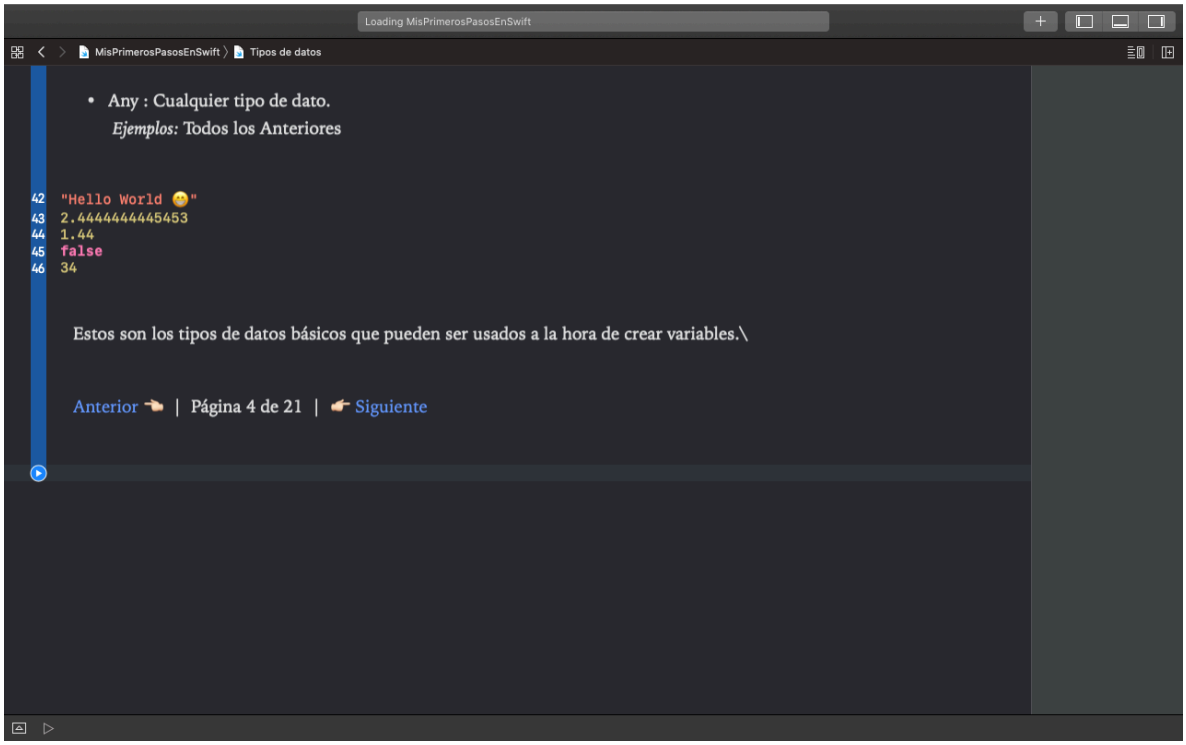

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Playground
¿Qué es swift?
Swift es un lenguaje de programación creado por Apple para el desarrollo de aplicaciones en iOS, OS X, Apple TV y Watch OS.
Swift es gratis y de código abierto.
¿Qué es un playground?
Un playground es un lugar donde puedes jugar, a la vez, experimentar con el código y ver resultados al instante sin ningún tipo de interrupciones. A continuación, se presentan sus funcionalidades en el código:
14

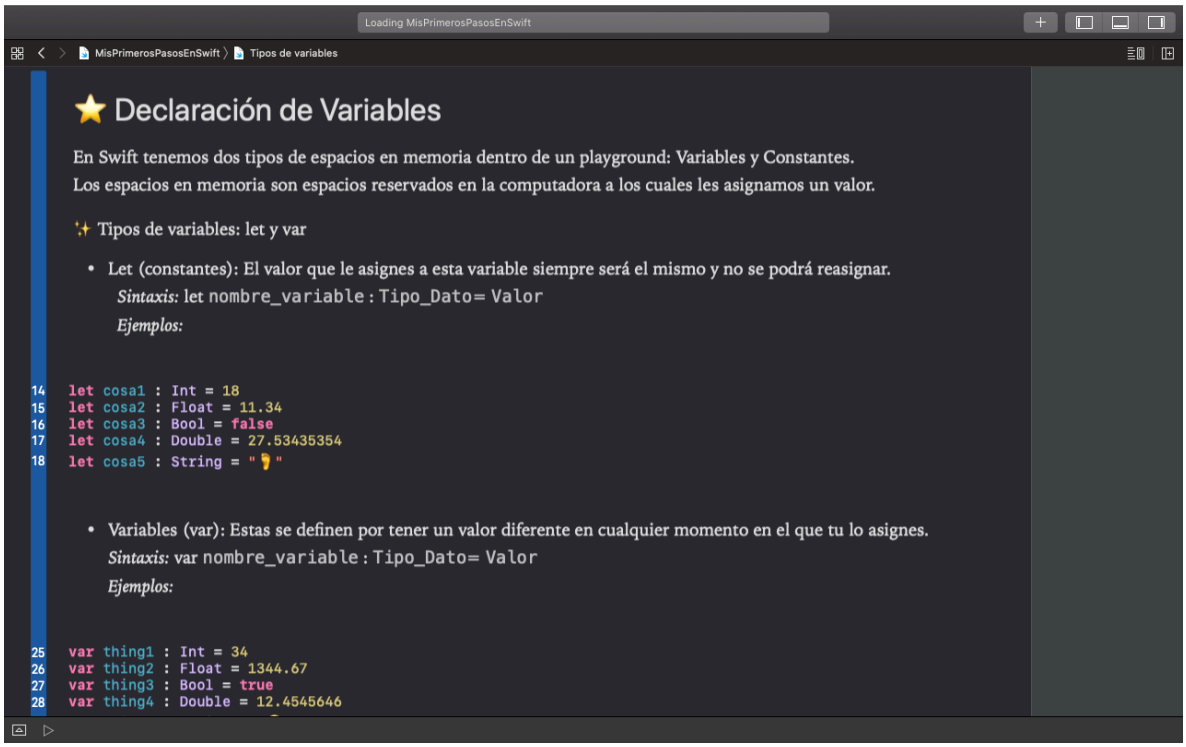

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Playground
•Sección de Codificación
Aquí se encuentra el código editable.
1+1*9+10
•Barra lateral de resultados
A medida que se agrega código o se cambia, el Playground ejecuta el código constantemente y actualiza los datos en la barra lateral, no del resultado final de todas las líneas de instrucciones escritas en la sección de codificación, sino por cada línea de código. La barra lateral de resultados es un apoyo al momento de codificar las instrucciones para llegar a un resultado final exitosamente.A continuación, se muestra un ejemplo:\
1+1
3 + 5
500 + 3
4 - 3
•Consola
Es un centro de mensajes que muestra detalles sobre la forma en que se está ejecutando un programa. A continuación se muestra un ejemplo:\
var saludo = "Hola Mundo 😌"
print(saludo)


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Playground
En la consola se puede imprimir con la instrucción print cualquier resultado que se haya querido mostrar con las instrucciones del código editable, confirmando los resultados ya mostrados en la barra lateral de resultados.
•Comentarios
Las notas dentro del código se denominan comentarios. El Playground ignora los comentarios, y éstos no afectan la ejecución del código. Un comentario comienza con dos barras // y con /**/ como se muestra a continuación:
19 //Ese número representa mi edad
/*
1+1
*/
•Errores
Cuando el playground encuentra un error en el código, detiene su ejecución y no se muestran los resultados en la barra lateral de resultados. Esta página del playground, por ejemplo, tiene un error.
//Hola Mundo 😌
❗ Observa que se resalta la línea de código en rojo en donde se encuentra dicho error y justo a la derecha aparece una descripción del error. Tampoco se muestran los resultados en la barra lateral.
Anterior 👈 | Página 3 de 21 | 👉 Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Tipos de datos
⭐ Tipos de datos
En la vida real, usar el tipo de objeto correcto es muy importante.
Si estás jugando ping pong, es importante usar una pelota en vez de cualquier otro objeto, como un huevo.Por otra parte, tener una docena de pelotas no es muy útil si estás tratando de cocinar un omelet.
Cada tipo de objeto tiene diferentes propiedades y comportamientos. En programación, los tipos de objetos funcionan de manera muy similar.\ En Swift cada objeto tiene su propio conjunto de propiedades y comportamientos.
A continuación, se muestra una lista de los tipos de datos que existen en Swift:
10
• Int : Se utiliza para números enteros
Ejemplos:
14 1212312313
15 34
• Bool : Un booleano es un indicador de 0 (false) o 1 (true).
Ejemplos:
true


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Tipos de datos
19 true
20 false
• Float : Números decimales con hasta 6 decimales de precisión.
Ejemplos:
24 1.2
25 1.33
26 1.44
• Double : Números decimales con con hasta 15 decimales de precisión.
Ejemplos:
30 1.33333333333
31 2.444444445453
32 445345.534534534
• String : Cadena de caracteres y Emojis Ejemplos:
36 "Hola Mundo"
37 "Hello World 😁"
38 "😁"


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift 〉 Tipos de datos
• Any : Cualquier tipo de dato.
Ejemplos: Todos los Anteriores
"Hello World 😬"
2.4444444445453
1.44
false
34
Estos son los tipos de datos básicos que pueden ser usados a la hora de crear variables.\
Anterior 👈 | Página 4 de 21 | 👉 Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift 〉 Tipos de variables
⭐ Declaración de Variables
En Swift tenemos dos tipos de espacios en memoria dentro de un playground: Variables y Constantes.
Los espacios en memoria son espacios reservados en la computadora a los cuales les asignamos un valor.
✨ Tipos de variables: let y var
• Let (constantes): El valor que le asignes a esta variable siempre será el mismo y no se podrá reasignar.
Sintaxis: let nombre_variable : Tipo_Dato= Valor
Ejemplos:
let cosa1 : Int = 18
let cosa2 : Float = 11.34
let cosa3 : Bool = false
let cosa4 : Double = 27.53435354
let cosa5 : String = "👇"
• Variables (var): Estas se definen por tener un valor diferente en cualquier momento en el que tu lo asignes.
Sintaxis: var nombre_variable : Tipo_Dato= Valor
Ejemplos:
var thing1 : Int = 34
var thing2 : Float = 1344.67
var thing3 : Bool = true
var thing4 : Double = 12.4545646

Loading MisPrimerosPasosEnSwift

MisPrimerosPasosEnSwift 〉 Tipos de variables

```
29 var thing5 : String = "🥺"
```

> **Note**
> Los ":" le indican a la computadora reservar un espacio de memoria del tipo de dato que le asignes

⭐ Operaciones con var y let

Podemos realizar operaciones asignando nuevas variables de la siguiente manera.

> **Note**
> La función `print(nombre_variable)` permite imprimir uno o varios valores en la consola.

Ejemplo:

```
43 print(cosa1, cosa2, cosa3, cosa4, cosa5)
44 print(thing1, thing2, thing3, thing4, thing5)
45 var suma : Int = cosa1 + thing1
46 print(suma)
47
48 var resta : Float = cosa2 - thing2
49 print(resta)
50
51 var suma1 : String = cosa5 + thing5
52 print(suma1)
53
54 var multiplicacion : Double = cosa4 * thing4
55 print(multiplicacion)
56
```

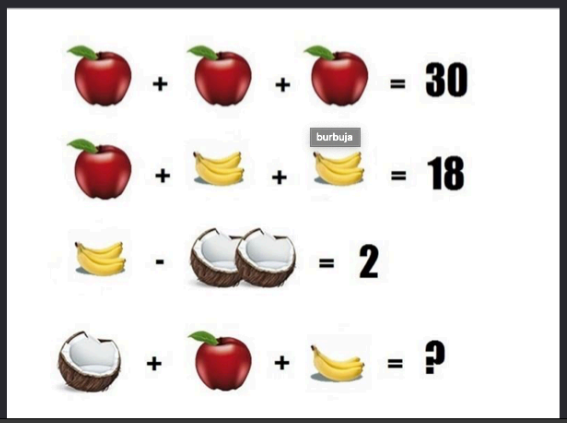
Loading MisPrimerosPasosEnSwift

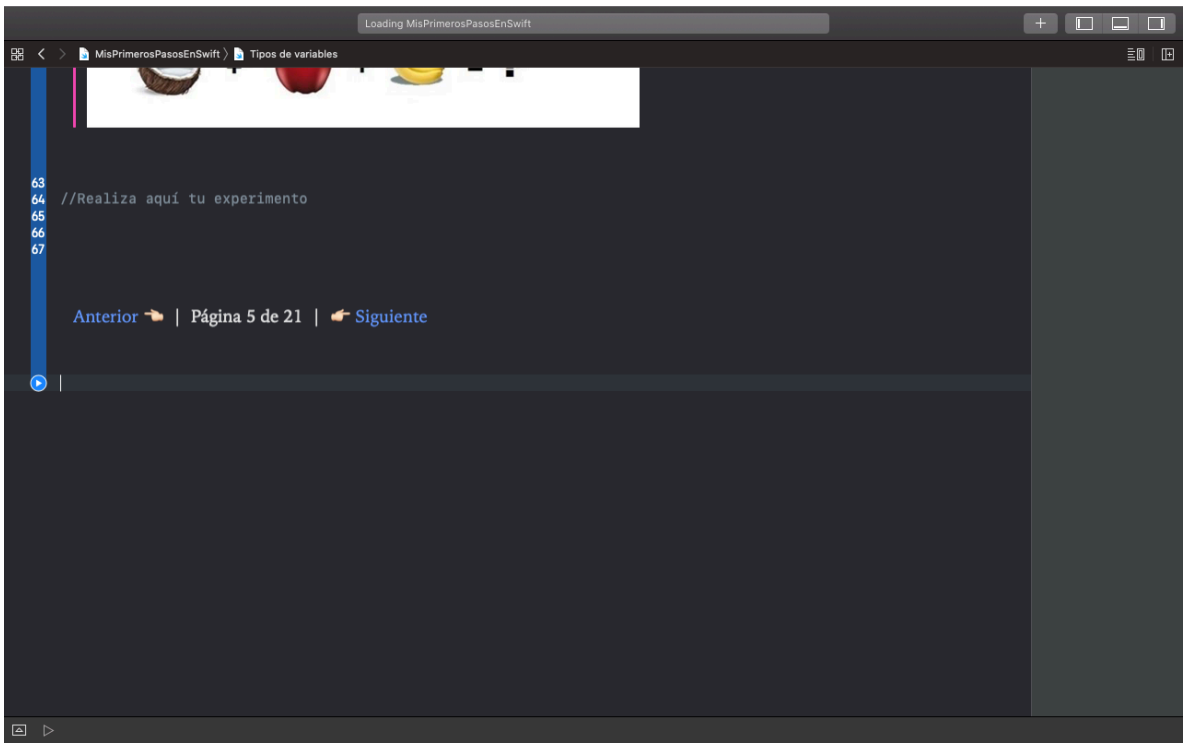
MisPrimerosPasosEnSwift 〉 Tipos de variables

> **Important**
> No se pueden realizar operaciones con diferentes tipos de datos.

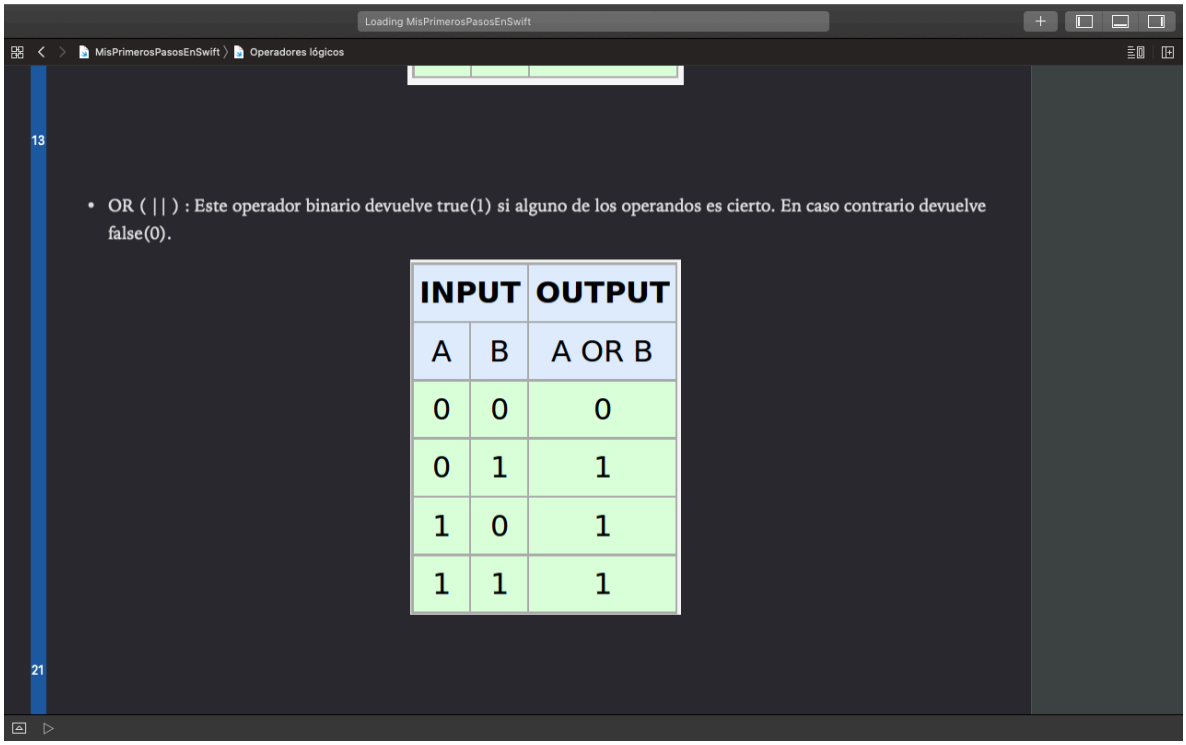
59

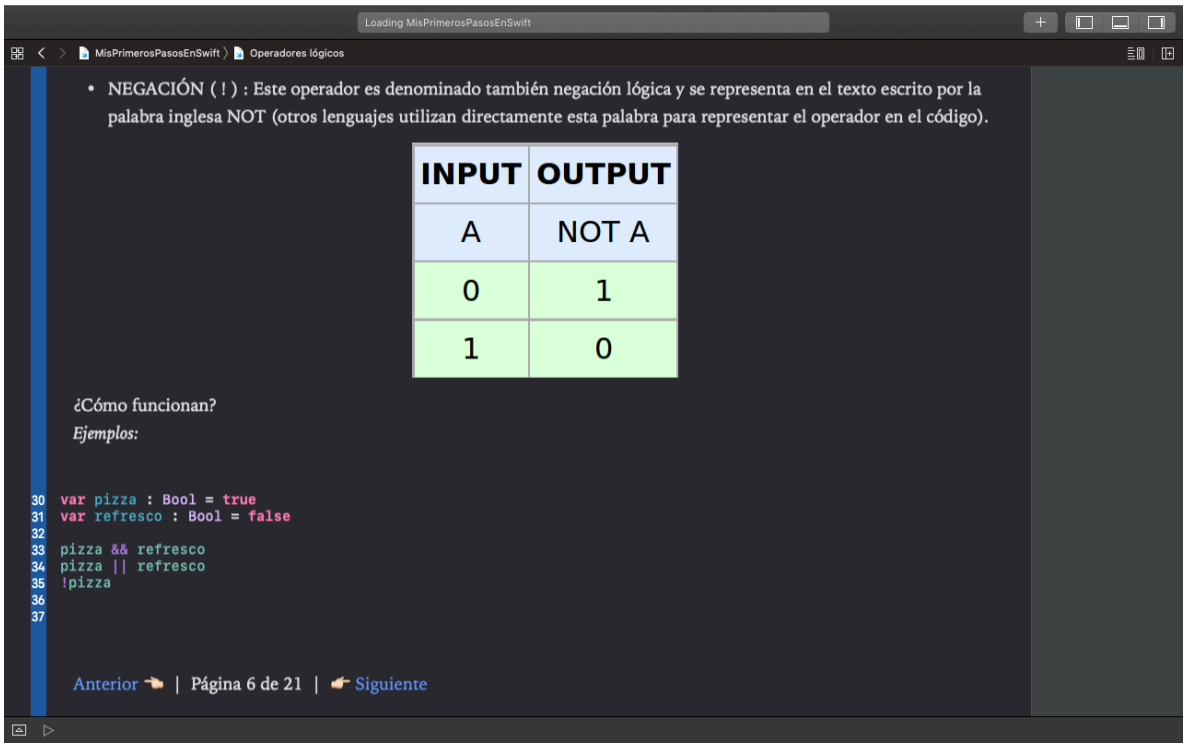
> **Experiment**
> 🔬 Realiza el siguiente experimiento con variables:

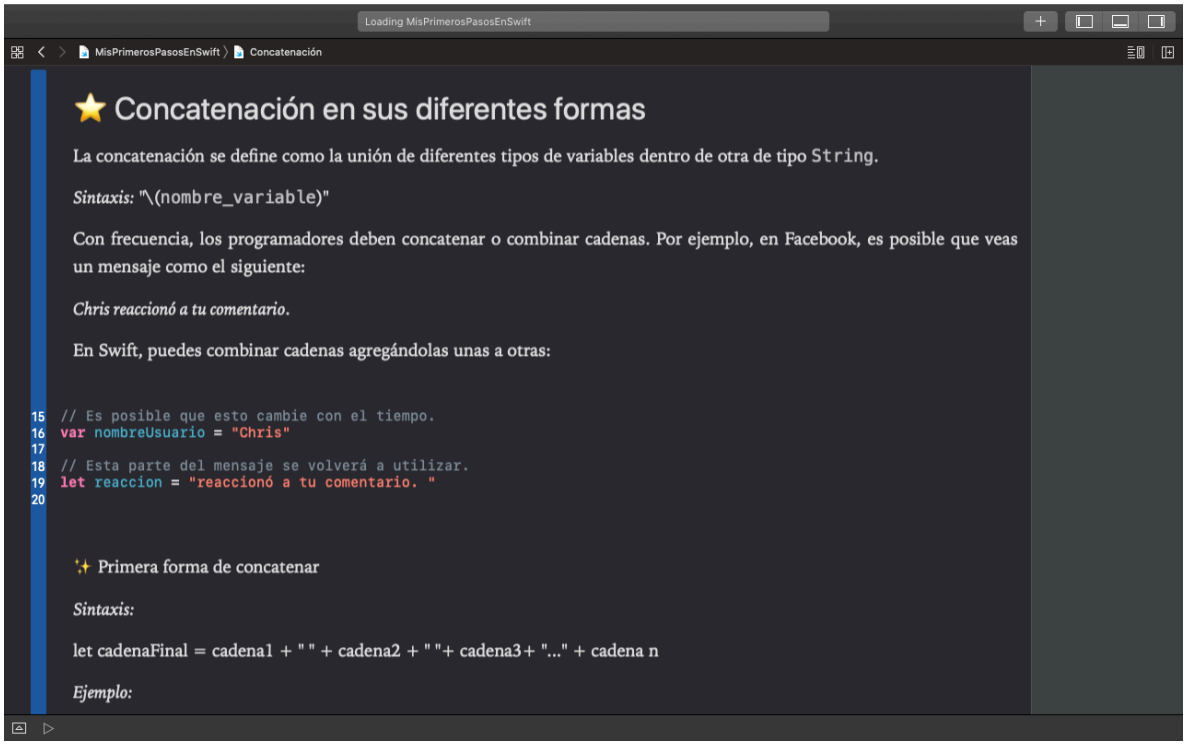


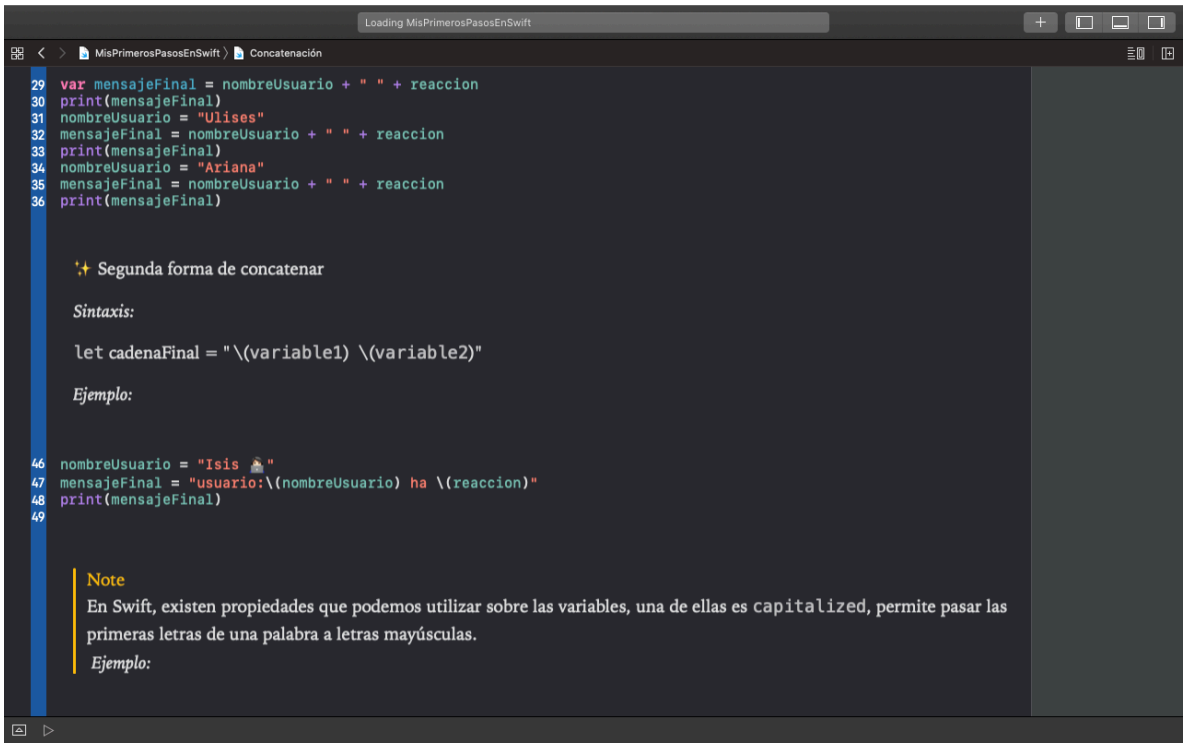


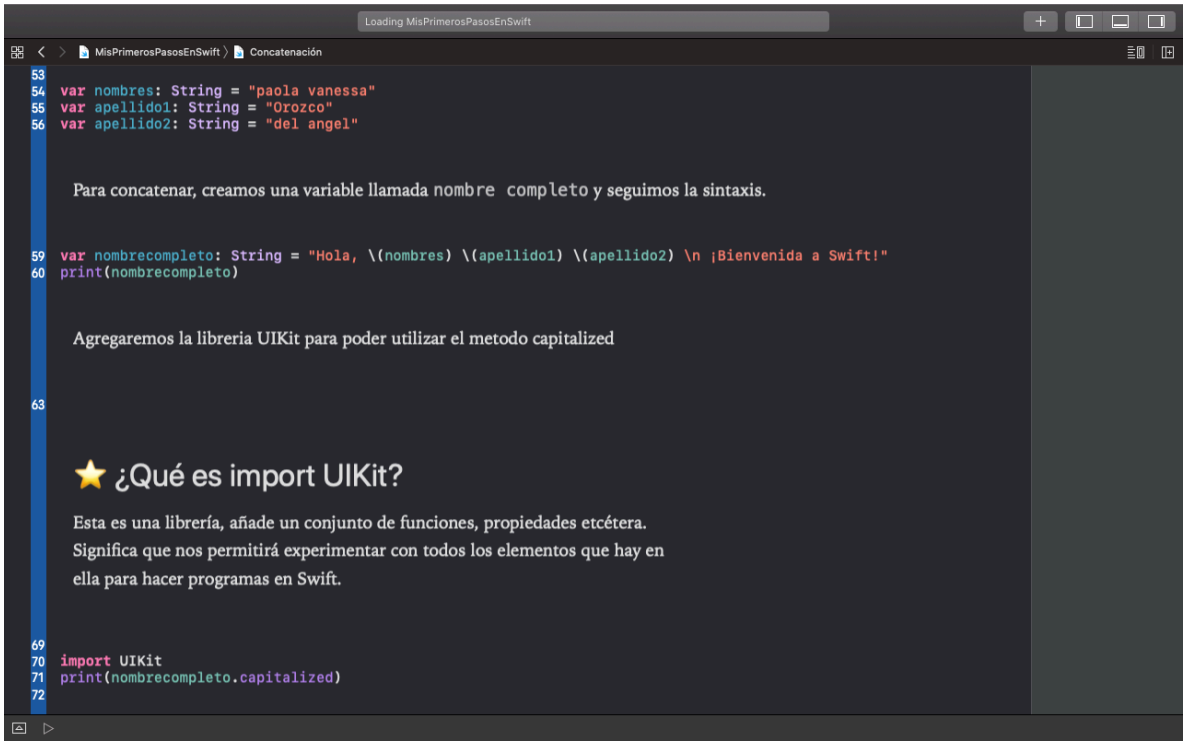
INPUT		OUTPUT
A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

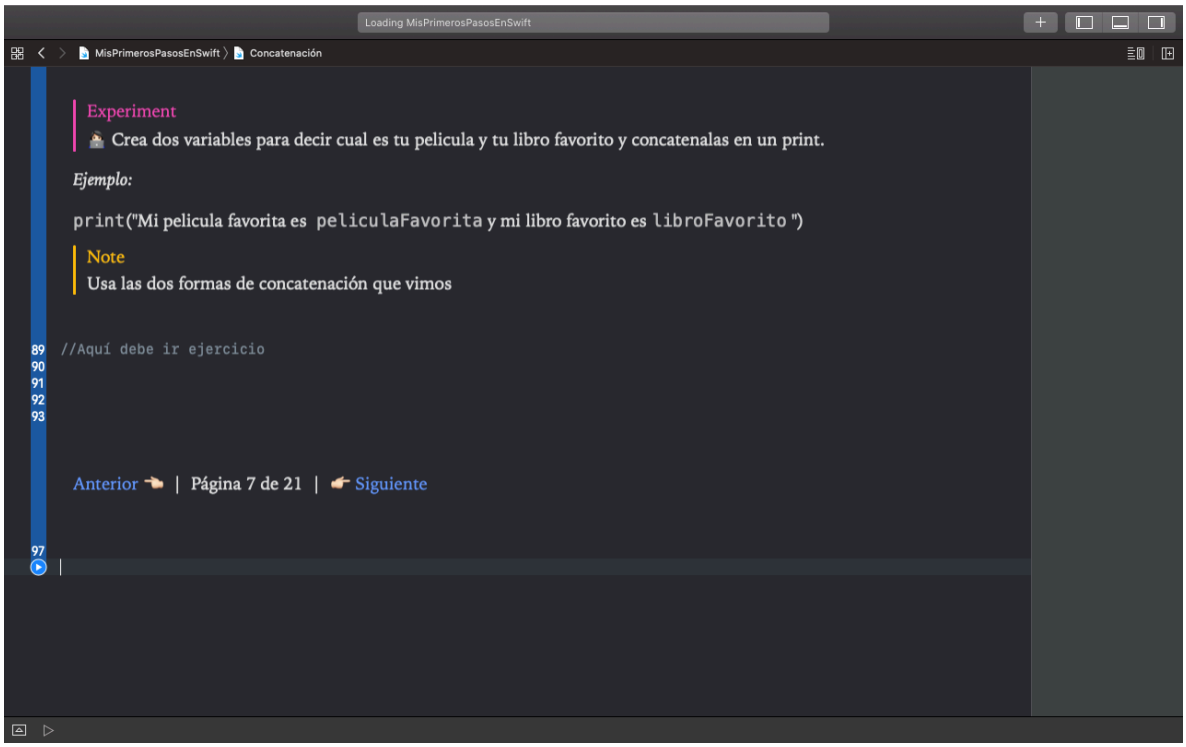

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Operadores lógicos
13
• OR (||) : Este operador binario devuelve true(1) si alguno de los operandos es cierto. En caso contrario devuelve false(0).
INPUT
OUTPUT
A
B
A OR B
0 0 0
0 1 1
1 0 1
1 1 1
21

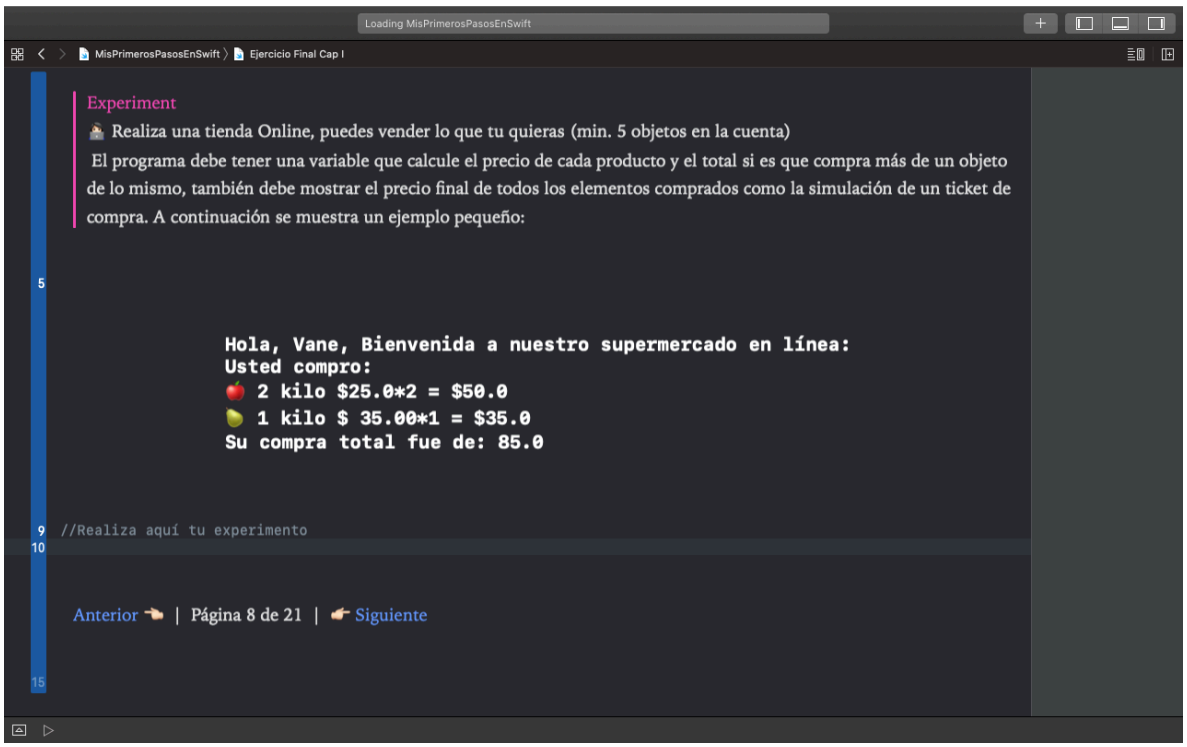

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Operadores lógicos
• NEGACIÓN (!) : Este operador es denominado también negación lógica y se representa en el texto escrito por la palabra inglesa NOT (otros lenguajes utilizan directamente esta palabra para representar el operador en el código).
INPUT
OUTPUT
A
NOT A
0 1
1 0
¿Cómo funcionan?
Ejemplos:
30 var pizza : Bool = true
31 var refresco : Bool = false
32
33 pizza && refresco
34 pizza || refresco
35 !pizza
36
37
Anterior | Página 6 de 21 | Siguiente

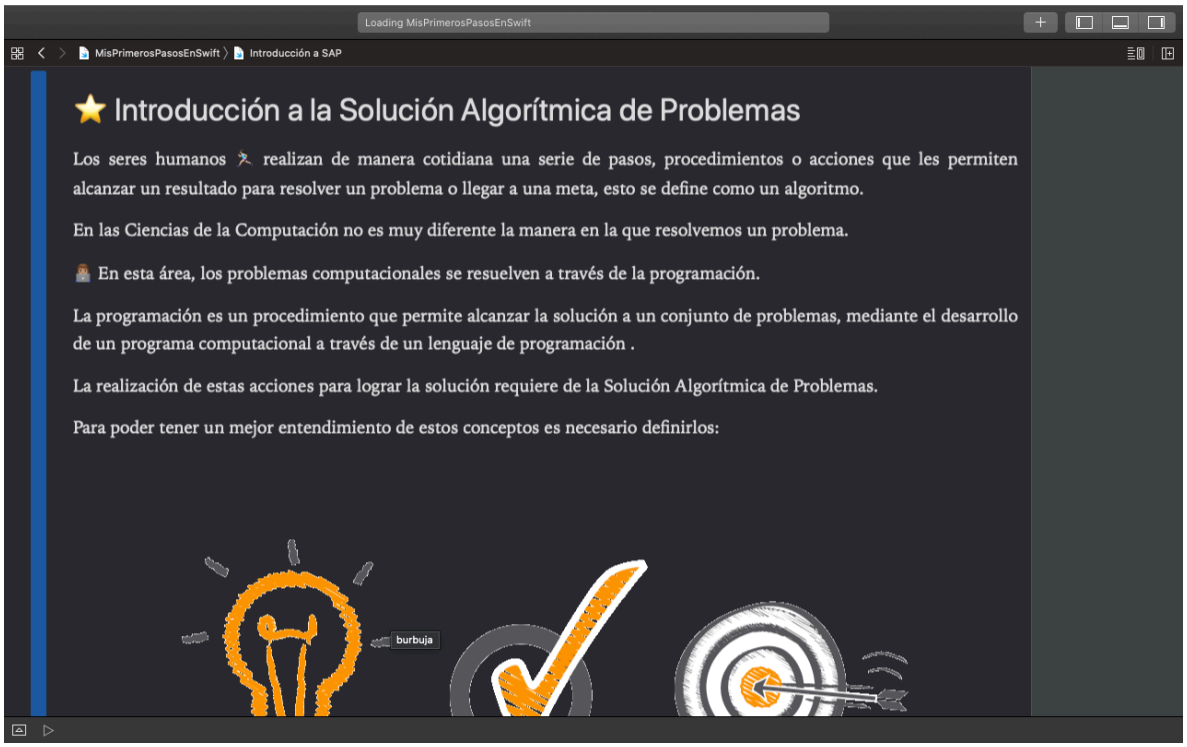

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Concatenación
⭐ Concatenación en sus diferentes formas
La concatenación se define como la unión de diferentes tipos de variables dentro de otra de tipo String.
Sintaxis: "\(nombre_variable)"
Con frecuencia, los programadores deben concatenar o combinar cadenas. Por ejemplo, en Facebook, es posible que veas un mensaje como el siguiente:
Chris reaccionó a tu comentario.
En Swift, puedes combinar cadenas agregándolas unas a otras:
15 // Es posible que esto cambie con el tiempo.
16 var nombreUsuario = "Chris"
17
18 // Esta parte del mensaje se volverá a utilizar.
19 let reaccion = "reaccionó a tu comentario. "
20
✨ Primera forma de concatenar
Sintaxis:
let cadenaFinal = cadena1 + " " + cadena2 + " "+ cadena3+ "..." + cadena n
Ejemplo:


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Concatenación
29 var mensajeFinal = nombreUsuario + " " + reaccion
30 print(mensajeFinal)
31 nombreUsuario = "Ulises"
32 mensajeFinal = nombreUsuario + " " + reaccion
33 print(mensajeFinal)
34 nombreUsuario = "Ariana"
35 mensajeFinal = nombreUsuario + " " + reaccion
36 print(mensajeFinal)
✨ Segunda forma de concatenar
Sintaxis:
let cadenaFinal = "\(variable1) \(variable2)"
Ejemplo:
46 nombreUsuario = "Isis 👩🏻‍🚀"
47 mensajeFinal = "usuario:\(nombreUsuario) ha \(reaccion)"
48 print(mensajeFinal)
49
Note
En Swift, existen propiedades que podemos utilizar sobre las variables, una de ellas es capitalized, permite pasar las primeras letras de una palabra a letras mayúsculas.
Ejemplo:

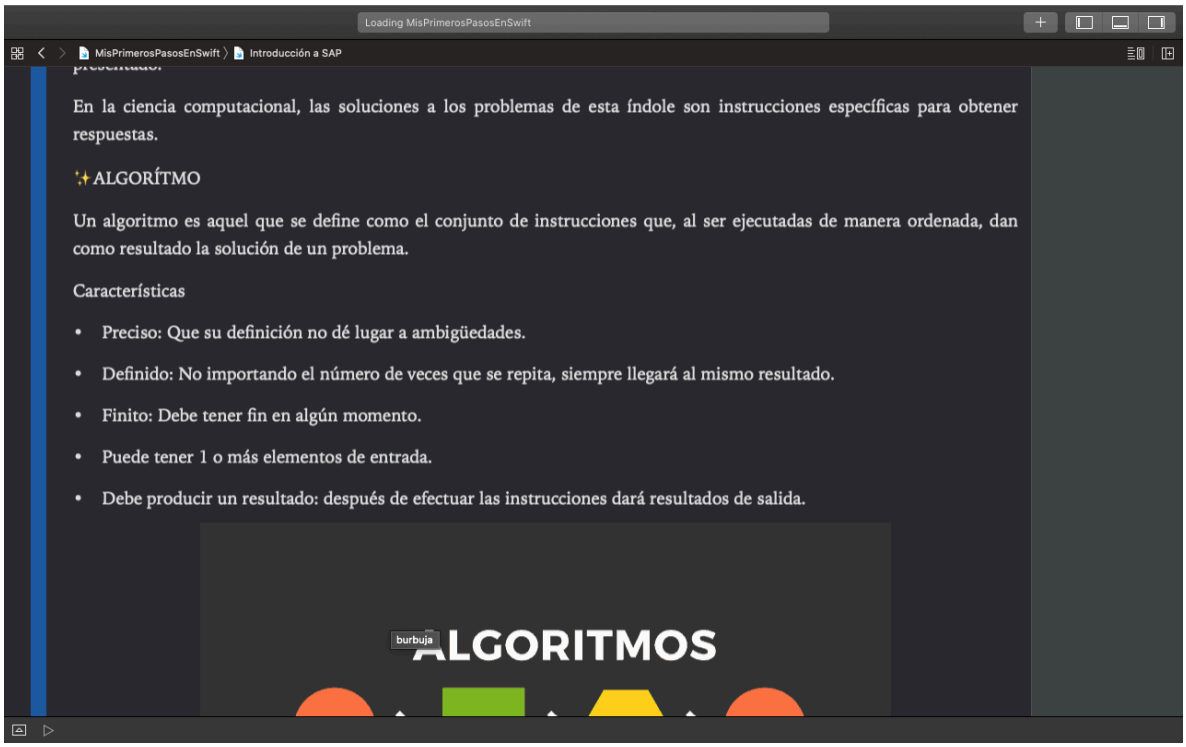

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Concatenación
var nombres: String = "paola vanessa"
var apellido1: String = "Orozco"
var apellido2: String = "del angel"
Para concatenar, creamos una variable llamada nombre completo y seguimos la sintaxis.
var nombrecompleto: String = "Hola, \(nombres) \(apellido1) \(apellido2) \n ¡Bienvenida a Swift!"
print(nombrecompleto)
Agregaremos la libreria UIKit para poder utilizar el metodo capitalized
⭐ ¿Qué es import UIKit?
Esta es una librería, añade un conjunto de funciones, propiedades etcétera.
Significa que nos permitirá experimentar con todos los elementos que hay en
ella para hacer programas en Swift.
import UIKit
print(nombrecompleto.capitalized)


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Concatenación
Experiment
Crea dos variables para decir cual es tu pelicula y tu libro favorito y concatenalas en un print.
Ejemplo:
print("Mi pelicula favorita es peliculaFavorita y mi libro favorito es libroFavorito ")
Note
Usa las dos formas de concatenación que vimos
//Aquí debe ir ejercicio
Anterior | Página 7 de 21 | Siguiente

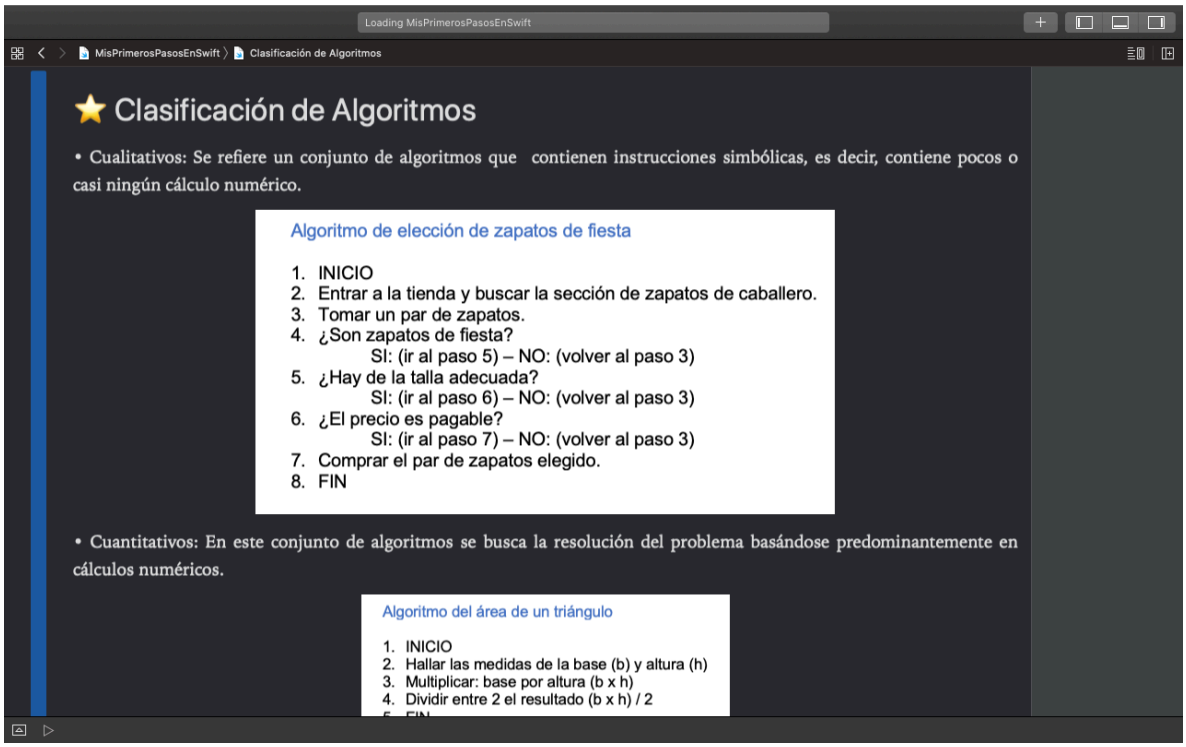

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Ejercicio Final Cap I
Experiment
Realiza una tienda Online, puedes vender lo que tu quieras (min. 5 objetos en la cuenta)
El programa debe tener una variable que calcule el precio de cada producto y el total si es que compra más de un objeto de lo mismo, también debe mostrar el precio final de todos los elementos comprados como la simulación de un ticket de compra. A continuación se muestra un ejemplo pequeño:
Hola, Vane, Bienvenida a nuestro supermercado en línea:
Usted compro:
2 kilo $25.0*2 = $50.0
1 kilo $ 35.00*1 = $35.0
Su compra total fue de: 85.0
//Realiza aquí tu experimento
Anterior | Página 8 de 21 | Siguiente

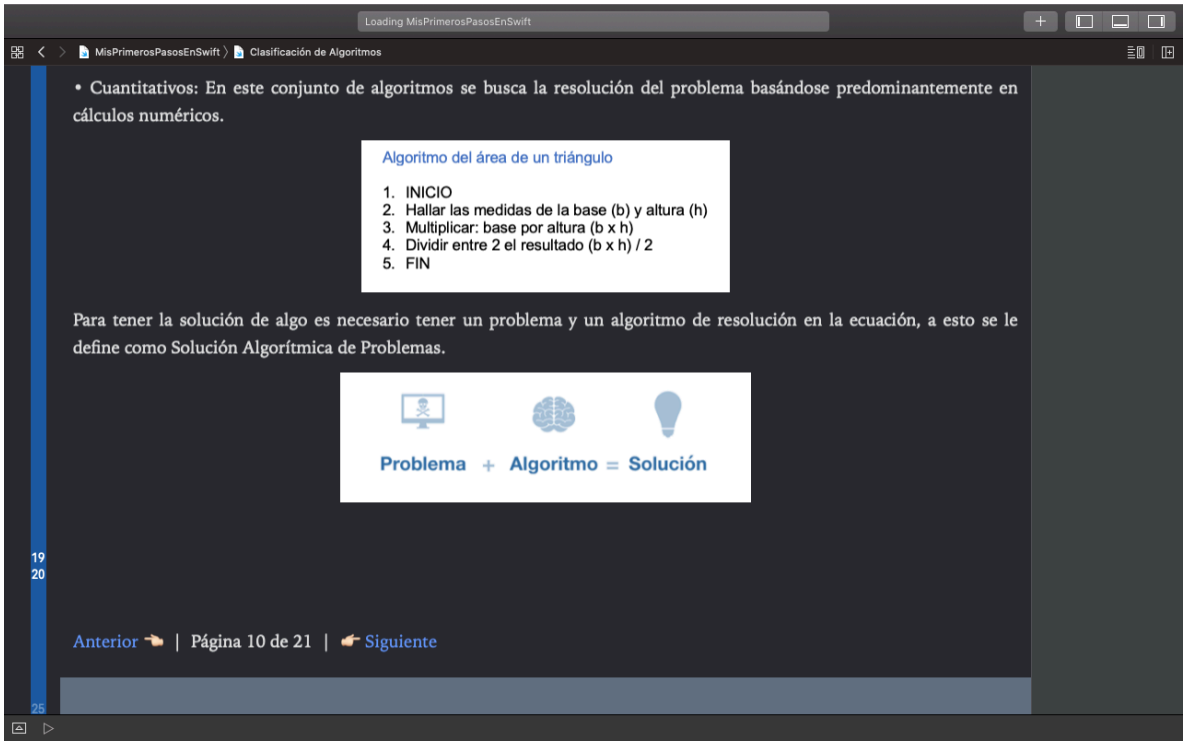

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Introducción a SAP
Introducción a la Solución Algorítmica de Problemas
Los seres humanos realizan de manera cotidiana una serie de pasos, procedimientos o acciones que les permiten alcanzar un resultado para resolver un problema o llegar a una meta, esto se define como un algoritmo.
En las Ciencias de la Computación no es muy diferente la manera en la que resolvemos un problema.
En esta área, los problemas computacionales se resuelven a través de la programación.
La programación es un procedimiento que permite alcanzar la solución a un conjunto de problemas, mediante el desarrollo de un programa computacional a través de un lenguaje de programación .
La realización de estas acciones para lograr la solución requiere de la Solución Algorítmica de Problemas.
Para poder tener un mejor entendimiento de estos conceptos es necesario definirlos:
burbuja

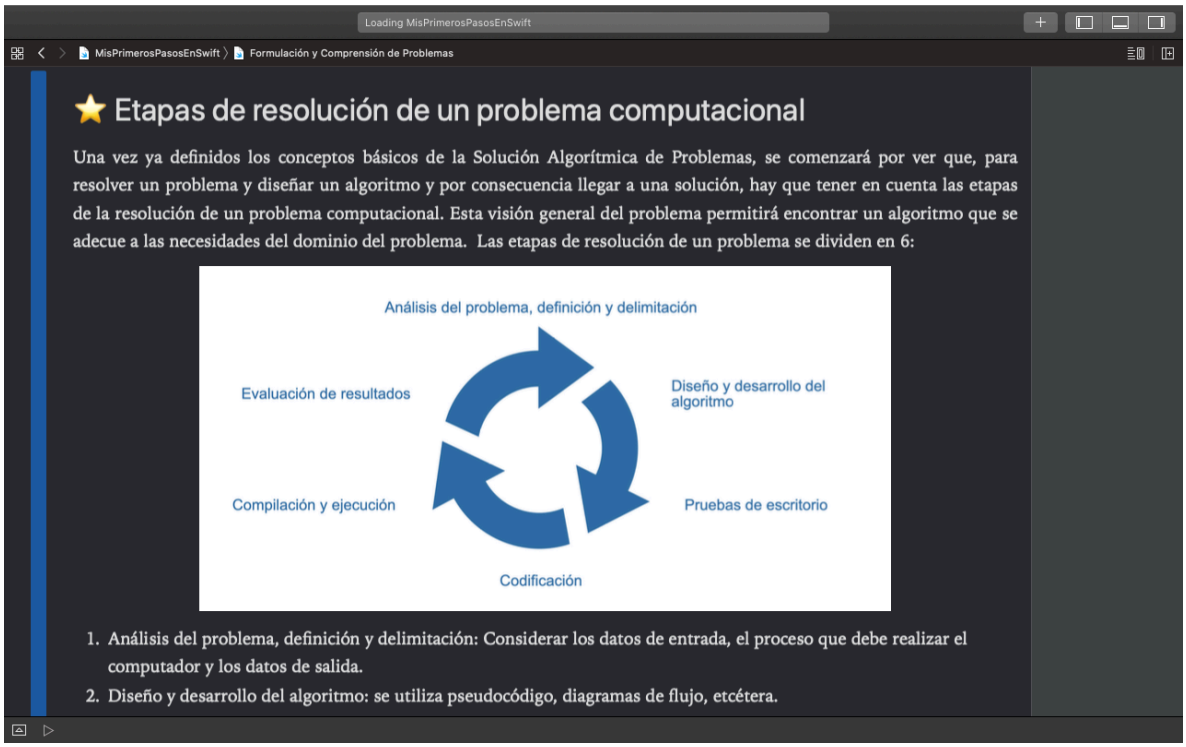

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Introducción a SAP
✨PROBLEMA
Comúnmente, se puede definir a un problema como un conflicto que se presenta, un inconveniente para alcanzar objetivos en distintos ámbitos, desarrollando la necesidad de encontrar una solución, siguiendo un procedimiento.
En el mundo computacional los tipos de problemas más importantes son de: Clasificación, Búsqueda, Procesamiento de cadenas, Problemas gráficos, Problemas combinatorios, Problemas geométricos y Problemas numéricos, entre otros.
✨SOLUCIÓN

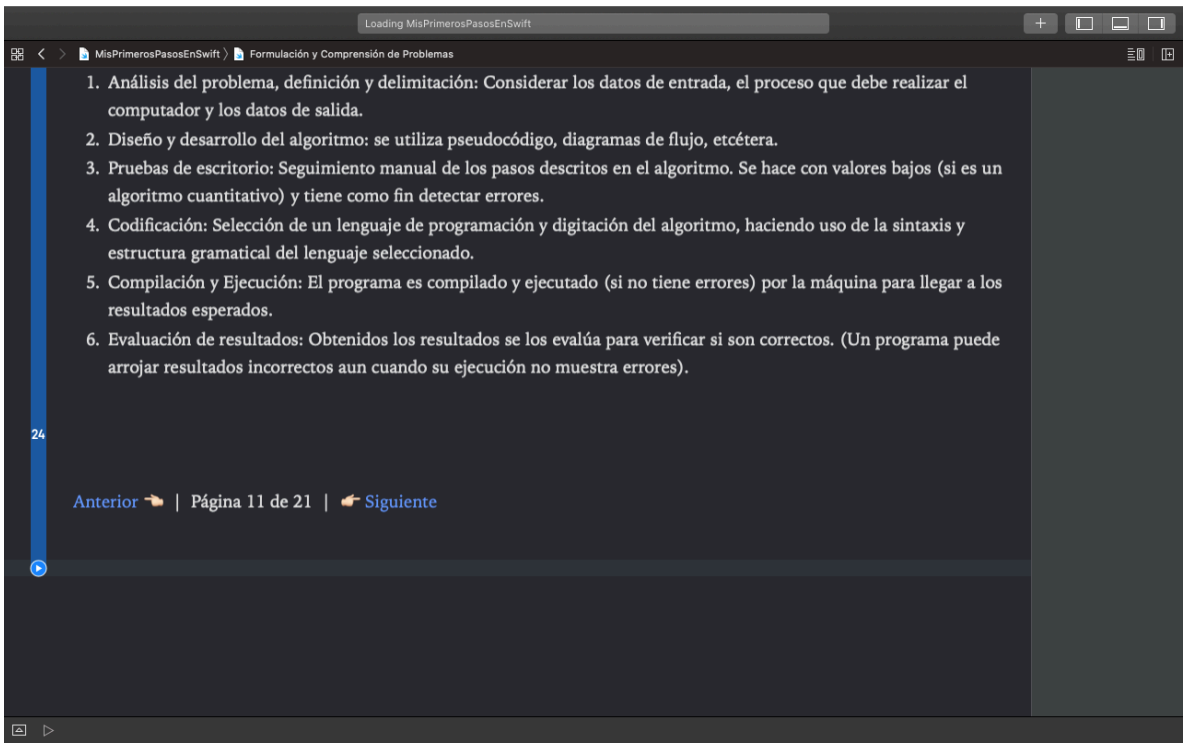

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Introducción a SAP
En la ciencia computacional, las soluciones a los problemas de esta índole son instrucciones específicas para obtener respuestas.
✨ALGORÍTMO
Un algoritmo es aquel que se define como el conjunto de instrucciones que, al ser ejecutadas de manera ordenada, dan como resultado la solución de un problema.
Características
• Preciso: Que su definición no dé lugar a ambigüedades.
• Definido: No importando el número de veces que se repita, siempre llegará al mismo resultado.
• Finito: Debe tener fin en algún momento.
• Puede tener 1 o más elementos de entrada.
• Debe producir un resultado: después de efectuar las instrucciones dará resultados de salida.
burbuja
ALGORITMOS

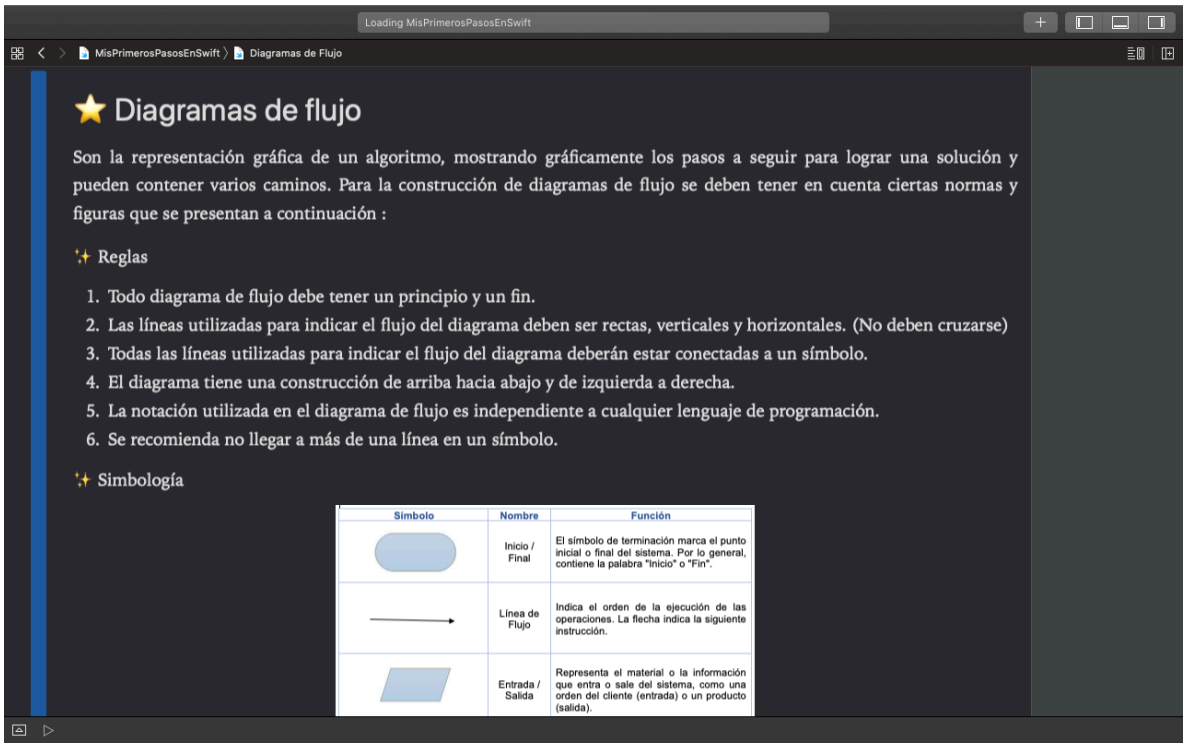

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Introducción a SAP
• Puede tener 1 o más elementos de entrada.
• Debe producir un resultado: después de efectuar las instrucciones dará resultados de salida.
ALGORITMOS
59
Anterior | Página 9 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Clasificación de Algoritmos
⭐ Clasificación de Algoritmos
• Cualitativos: Se refiere un conjunto de algoritmos que contienen instrucciones simbólicas, es decir, contiene pocos o casi ningún cálculo numérico.
Algoritmo de elección de zapatos de fiesta
1. INICIO
2. Entrar a la tienda y buscar la sección de zapatos de caballero.
3. Tomar un par de zapatos.
4. ¿Son zapatos de fiesta?
SI: (ir al paso 5) – NO: (volver al paso 3)
5. ¿Hay de la talla adecuada?
SI: (ir al paso 6) – NO: (volver al paso 3)
6. ¿El precio es pagable?
SI: (ir al paso 7) – NO: (volver al paso 3)
7. Comprar el par de zapatos elegido.
8. FIN
• Cuantitativos: En este conjunto de algoritmos se busca la resolución del problema basándose predominantemente en cálculos numéricos.
Algoritmo del área de un triángulo
1. INICIO
2. Hallar las medidas de la base (b) y altura (h)
3. Multiplicar: base por altura (b x h)
4. Dividir entre 2 el resultado (b x h) / 2


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Clasificación de Algoritmos
• Cuantitativos: En este conjunto de algoritmos se busca la resolución del problema basándose predominantemente en cálculos numéricos.
Algoritmo del área de un triángulo
1. INICIO
2. Hallar las medidas de la base (b) y altura (h)
3. Multiplicar: base por altura (b x h)
4. Dividir entre 2 el resultado (b x h) / 2
5. FIN
Para tener la solución de algo es necesario tener un problema y un algoritmo de resolución en la ecuación, a esto se le define como Solución Algorítmica de Problemas.
Problema + Algoritmo = Solución
Anterior | Página 10 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Formulación y Comprensión de Problemas
⭐ Etapas de resolución de un problema computacional
Una vez ya definidos los conceptos básicos de la Solución Algorítmica de Problemas, se comenzará por ver que, para resolver un problema y diseñar un algoritmo y por consecuencia llegar a una solución, hay que tener en cuenta las etapas de la resolución de un problema computacional. Esta visión general del problema permitirá encontrar un algoritmo que se adecue a las necesidades del dominio del problema. Las etapas de resolución de un problema se dividen en 6:
Análisis del problema, definición y delimitación
Diseño y desarrollo del algoritmo
Pruebas de escritorio
Codificación
Compilación y ejecución
Evaluación de resultados
1. Análisis del problema, definición y delimitación: Considerar los datos de entrada, el proceso que debe realizar el computador y los datos de salida.
2. Diseño y desarrollo del algoritmo: se utiliza pseudocódigo, diagramas de flujo, etcétera.


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Formulación y Comprensión de Problemas
1. Análisis del problema, definición y delimitación: Considerar los datos de entrada, el proceso que debe realizar el computador y los datos de salida.
2. Diseño y desarrollo del algoritmo: se utiliza pseudocódigo, diagramas de flujo, etcétera.
3. Pruebas de escritorio: Seguimiento manual de los pasos descritos en el algoritmo. Se hace con valores bajos (si es un algoritmo cuantitativo) y tiene como fin detectar errores.
4. Codificación: Selección de un lenguaje de programación y digitación del algoritmo, haciendo uso de la sintaxis y estructura gramatical del lenguaje seleccionado.
5. Compilación y Ejecución: El programa es compilado y ejecutado (si no tiene errores) por la máquina para llegar a los resultados esperados.
6. Evaluación de resultados: Obtenidos los resultados se los evalúa para verificar si son correctos. (Un programa puede arrojar resultados incorrectos aun cuando su ejecución no muestra errores).
24
Anterior | Página 11 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Diagramas de Flujo
⭐ Diagramas de flujo
Son la representación gráfica de un algoritmo, mostrando gráficamente los pasos a seguir para lograr una solución y pueden contener varios caminos. Para la construcción de diagramas de flujo se deben tener en cuenta ciertas normas y figuras que se presentan a continuación :
✨ Reglas
1. Todo diagrama de flujo debe tener un principio y un fin.
2. Las líneas utilizadas para indicar el flujo del diagrama deben ser rectas, verticales y horizontales. (No deben cruzarse)
3. Todas las líneas utilizadas para indicar el flujo del diagrama deberán estar conectadas a un símbolo.
4. El diagrama tiene una construcción de arriba hacia abajo y de izquierda a derecha.
5. La notación utilizada en el diagrama de flujo es independiente a cualquier lenguaje de programación.
6. Se recomienda no llegar a más de una línea en un símbolo.
✨ Simbología
Símbolo
Nombre
Función
Inicio / Final
El símbolo de terminación marca el punto inicial o final del sistema. Por lo general, contiene la palabra "Inicio" o "Fin".
Línea de Flujo
Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
Entrada / Salida
Representa el material o la información que entra o sale del sistema, como una orden del cliente (entrada) o un producto (salida).

Loading MisPrimerosPasosEnSwift

MisPrimerosPasosEnSwift 〉 Diagramas de Flujo

✨ Simbología

Símbolo	Nombre	Función
	Inicio / Final	El símbolo de terminación marca el punto inicial o final del sistema. Por lo general, contiene la palabra "Inicio" o "Fin".
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa el material o la información que entra o sale del sistema, como una orden del cliente (entrada) o un producto (salida).
	Entrada Manual	Representa un paso en el que se pide al usuario que introduzca la información manualmente.
	Proceso	Símbolo para representar un proceso. En su interior debe ir una asignación, operación, cambio de valor, etcétera.
	Decisión	Símbolo utilizado para representar una decisión. En su interior debe ir una condición la cual debe ser evaluada, dependiendo del resultado se toma el camino verdadero o falso. Este símbolo se utiliza en las estructuras selectivas si y si entonces/sino.
	Impresión	Símbolo que expresa escritura o impresión y se utiliza para imprimir un resultado.

Estos símbolos apoyan la construcción del pensamiento lógico en los alumnos al momento de la realización de un

Loading MisPrimerosPasosEnSwift

MisPrimerosPasosEnSwift 〉 Diagramas de Flujo

Estos símbolos apoyan la construcción del pensamiento lógico en los alumnos al momento de la realización de un algoritmo para llegar a la solución de un problema ya existente.

> Experiment
> Con los siguientes *Ejemplos Cualitativos y Cuantitativos* realiza la contrucción del diagrama de flujo de cada uno.

Ejemplo Cualitativo:

Recordando el algortimo de elección de zapatos de fiesta.

Algoritmo de elección de zapatos de fiesta

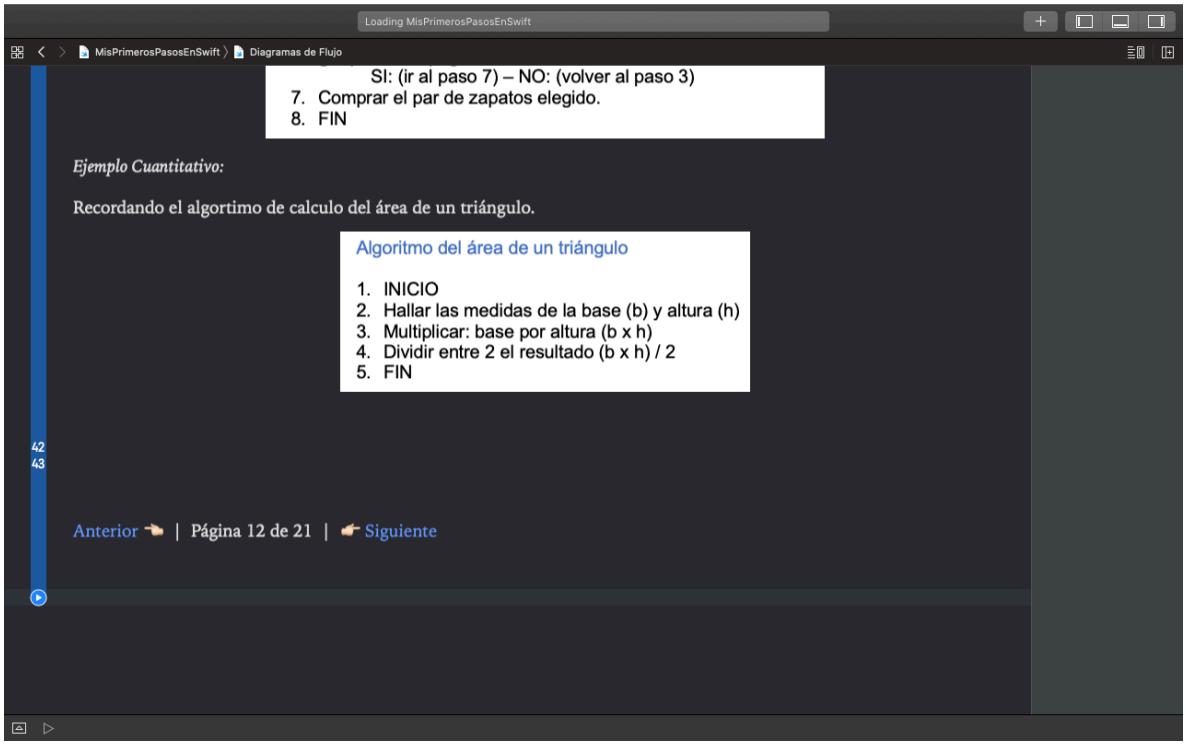
1. INICIO
2. Entrar a la tienda y buscar la sección de zapatos de caballero.
3. Tomar un par de zapatos.
4. ¿Son zapatos de fiesta?
   SI: (ir al paso 5) – NO: (volver al paso 3)
5. ¿Hay de la talla adecuada?
   SI: (ir al paso 6) – NO: (volver al paso 3)
6. ¿El precio es pagable?
   SI: (ir al paso 7) – NO: (volver al paso 3)
7. Comprar el par de zapatos elegido.
8. FIN

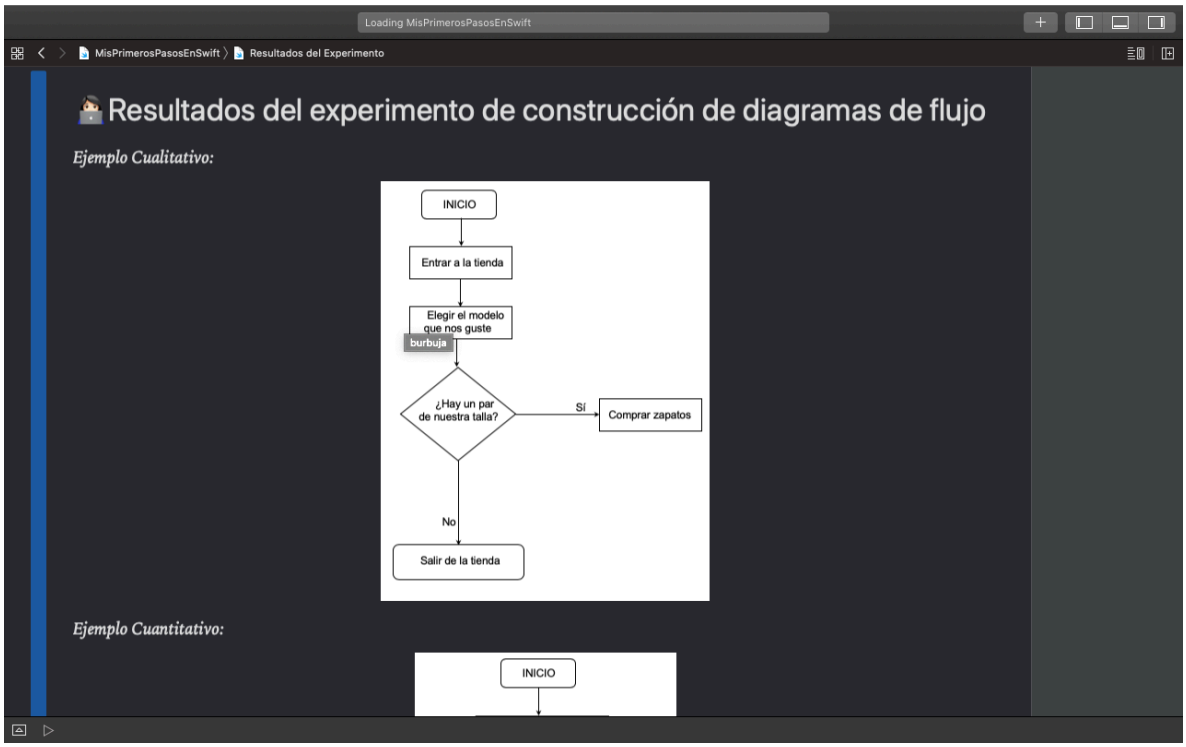
Ejemplo Cuantitativo:

Recordando el algortimo de calculo del área de un triángulo.

Algoritmo del área de un triángulo

1. INICIO

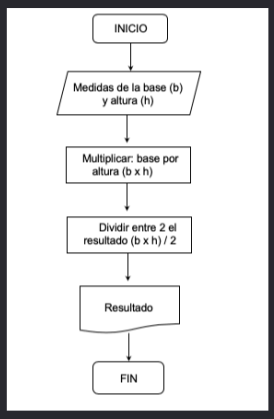

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Diagramas de Flujo
SI: (ir al paso 7) – NO: (volver al paso 3)
7. Comprar el par de zapatos elegido.
8. FIN
Ejemplo Cuantitativo:
Recordando el algortimo de calculo del área de un triángulo.
Algoritmo del área de un triángulo
1. INICIO
2. Hallar las medidas de la base (b) y altura (h)
3. Multiplicar: base por altura (b x h)
4. Dividir entre 2 el resultado (b x h) / 2
5. FIN
42
43
Anterior | Página 12 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Resultados del Experimento
Resultados del experimento de construcción de diagramas de flujo
Ejemplo Cualitativo:
INICIO
Entrar a la tienda
Elegir el modelo que nos guste
burbuja
¿Hay un par de nuestra talla?
Sí
Comprar zapatos
No
Salir de la tienda
Ejemplo Cuantitativo:
INICIO

Loading MisPrimerosPasosEnSwift

MisPrimerosPasosEnSwift 〉 Resultados del Experimento

Ejemplo Cuantitativo:



Anterior 👈 | Página 13 de 21 | 👉 Siguiente

Loading MisPrimerosPasosEnSwift

MisPrimerosPasosEnSwift 〉 Pseudocódigo

⭐ Pseudocódigo

El pseudocódigo es una forma de expresar los distintos pasos que va a realizar un programa, de la forma más parecida a un lenguaje de programación.

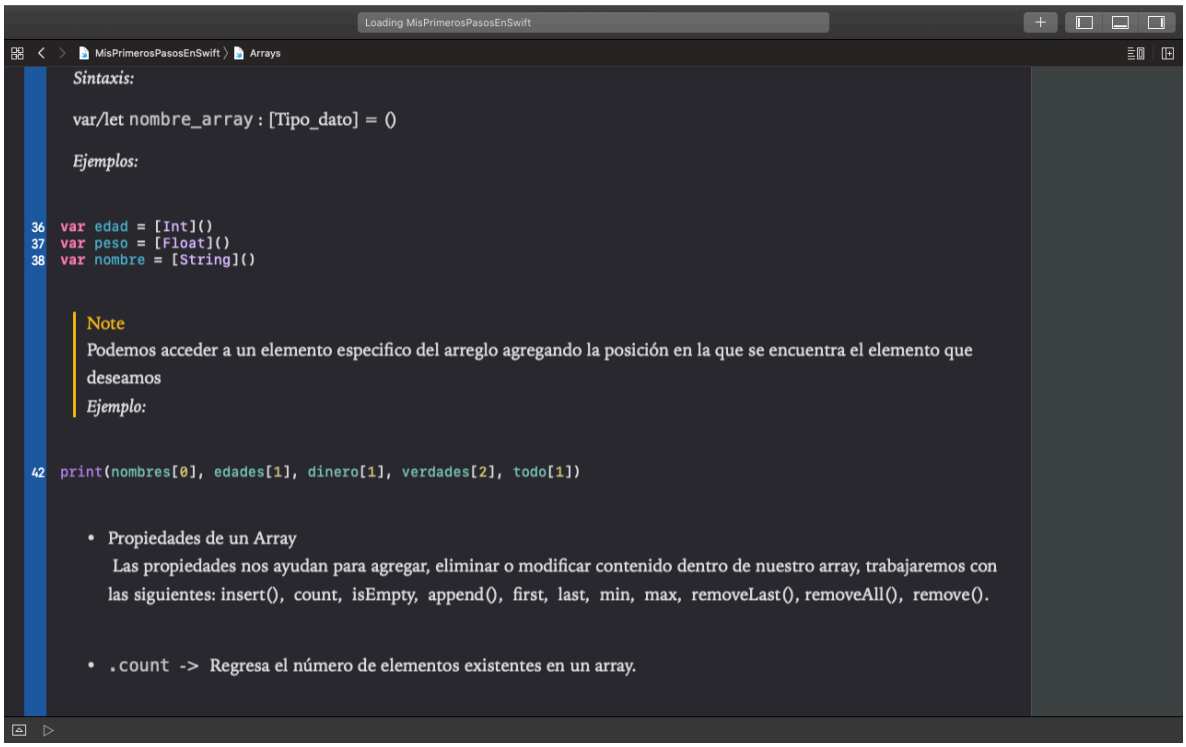
El pseudocódigo no puede ejecutarse en una computadora, ya que entonces dejaría de ser pseudocódigo, como su propio nombre lo indica, la palabra pseudo proviene del griego que significa falso, se trata de un código falso, escrito para que lo entienda el ser humano y no una máquina.

A continuación, se presenta un ejemplo de pseudocódigo del *Ejemplo Cuantitativo* del cálculo del área de un triángulo.

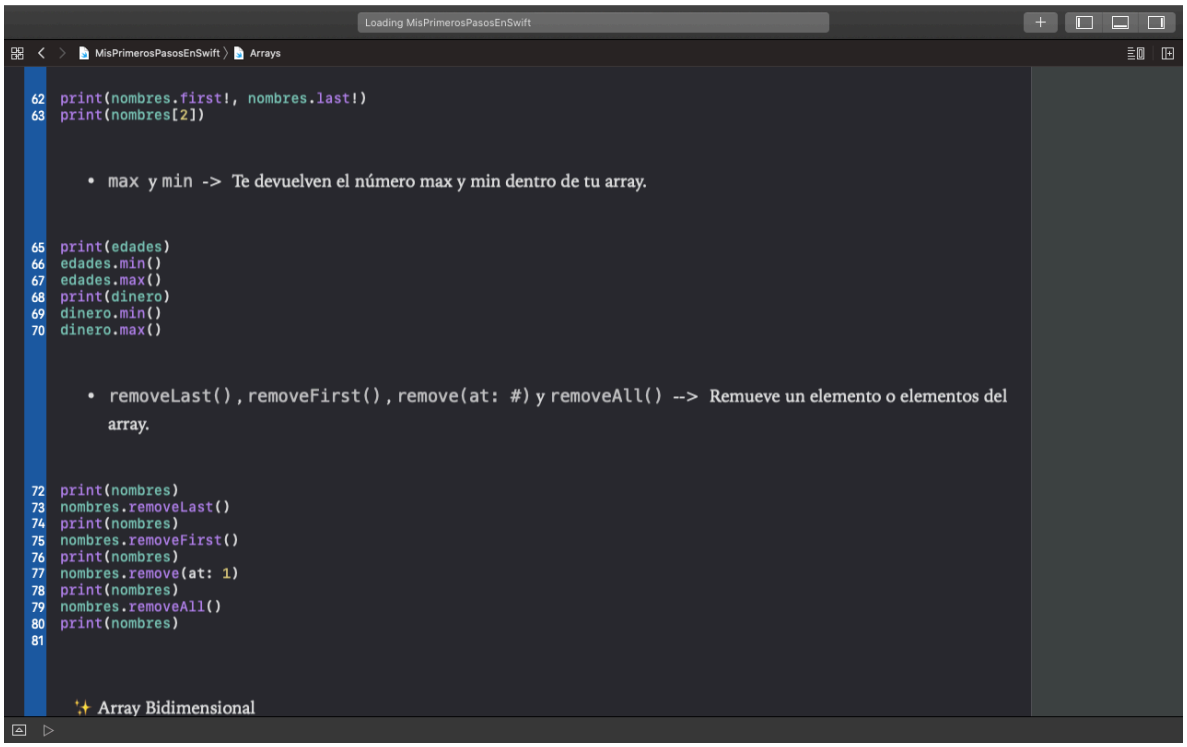
```
Pseudocódigo para calcular el área de un triángulo

Proceso Área
        Definir base (b), altura (h), área(a);
        área<-b * h;
        área<-área / 2;
        Escribir "El área del triángulo es:", área;
Fin Proceso Área
```

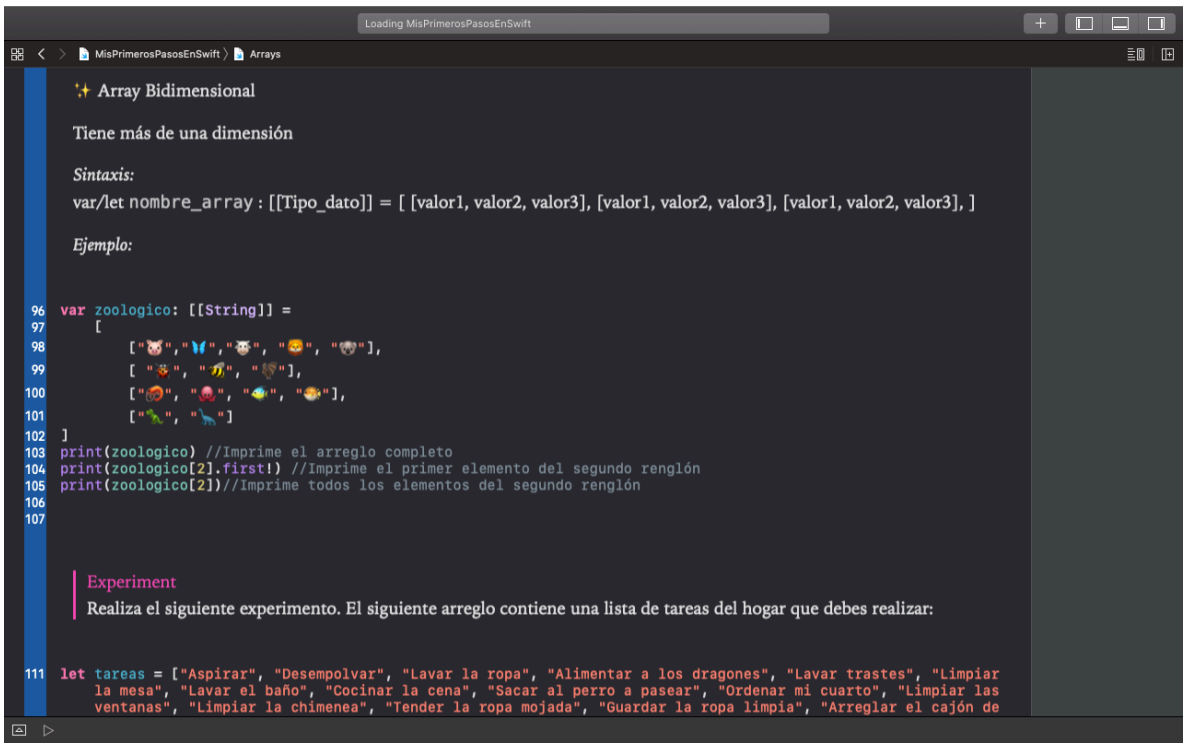
Anterior 👈 | Página 14 de 21 | 👉 Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Arrays
⭐ Colecciones de datos
En Swift tenemos 3 tipos de colecciones básicas: Arrays, Set y Diccionarios para alacenar colecciones de datos.
✨ Arrays
Usamos listas todo el tiempo. Podemos hacer listas de tareas pendientes, una lista de deseos, una lista de metas a corto y largo plazo, etc. En Swift, las listas se llaman Arrays/Arreglos.
Sintaxis:
var/let nombre_array : [Tipo_dato] = [valor1, valor2, valor3]
Ejemplos:
17 var nombres:[String] = ["Vane","Isis","Migue"]
18 print(nombres)
19 var edades:[Int] = [22,21,22]
20 print(edades)
21 var dinero:[Float] = [300.354, 764]
22 print(dinero)
23 var verdades:[Bool] = [true, true, false]
24 print(verdades)
25 var todo:[Any] = [1,"Hola",true]
26 print(todo)
También podemos crear arrays vacios, que no tengan un valor asignado.


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Arrays
Sintaxis:
var/let nombre_array : [Tipo_dato] = ()
Ejemplos:
36 var edad = [Int]()
37 var peso = [Float]()
38 var nombre = [String]()
Note
Podemos acceder a un elemento especifico del arreglo agregando la posición en la que se encuentra el elemento que deseamos
Ejemplo:
42 print(nombres[0], edades[1], dinero[1], verdades[2], todo[1])
• Propiedades de un Array
Las propiedades nos ayudan para agregar, eliminar o modificar contenido dentro de nuestro array, trabajaremos con las siguientes: insert(), count, isEmpty, append(), first, last, min, max, removeLast(), removeAll(), remove().
• .count -> Regresa el número de elementos existentes en un array.


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Arrays
print(nombres.count)
• .isEmpty -> Identifica si el arreglo: ¿Esta vacio?
print(nombres.isEmpty)
• ! -> si lo que queremos saber es: ¿Tiene algo?
print(!nombres.isEmpty)
• .append -> Agrega un elemento en el arreglo ya definido
print(dinero)
dinero.append(40)
print(dinero)
print(nombres)
nombres.append("Iván ")
print(nombres)
• .first -> Nos regresa el primer elemento de un array.
• .last -> Nos regresa el último elemento del array.


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Arrays
print(nombres.first!, nombres.last!)
print(nombres[2])
• max y min -> Te devuelven el número max y min dentro de tu array.
print(edades)
edades.min()
edades.max()
print(dinero)
dinero.min()
dinero.max()
• removeLast(), removeFirst(), remove(at: #) y removeAll() --> Remueve un elemento o elementos del array.
print(nombres)
nombres.removeLast()
print(nombres)
nombres.removeFirst()
print(nombres)
nombres.remove(at: 1)
print(nombres)
nombres.removeAll()
print(nombres)
✨ Array Bidimensional


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Arrays
✨ Array Bidimensional
Tiene más de una dimensión
Sintaxis:
var/let nombre_array : [[Tipo_dato]] = [[valor1, valor2, valor3], [valor1, valor2, valor3], [valor1, valor2, valor3],]
Ejemplo:
var zoologico: [[String]] =
[
["🐮","🦊","🐭", "🐵", "🐶"],
["🐯", "🐠", "🦚"],
["🐔", "🐙", "🌏", "🐲"],
["🦎", "🦕"]
]
print(zoologico) //Imprime el arreglo completo
print(zoologico[2].first!) //Imprime el primer elemento del segundo renglón
print(zoologico[2])//Imprime todos los elementos del segundo renglón
Experiment
Realiza el siguiente experimento. El siguiente arreglo contiene una lista de tareas del hogar que debes realizar:
let tareas = ["Aspirar", "Desempolvar", "Lavar la ropa", "Alimentar a los dragones", "Lavar trastes", "Limpiar la mesa", "Lavar el baño", "Cocinar la cena", "Sacar al perro a pasear", "Ordenar mi cuarto", "Limpiar las ventanas", "Limpiar la chimenea", "Tender la ropa mojada", "Guardar la ropa limpia", "Arreglar el cajón de

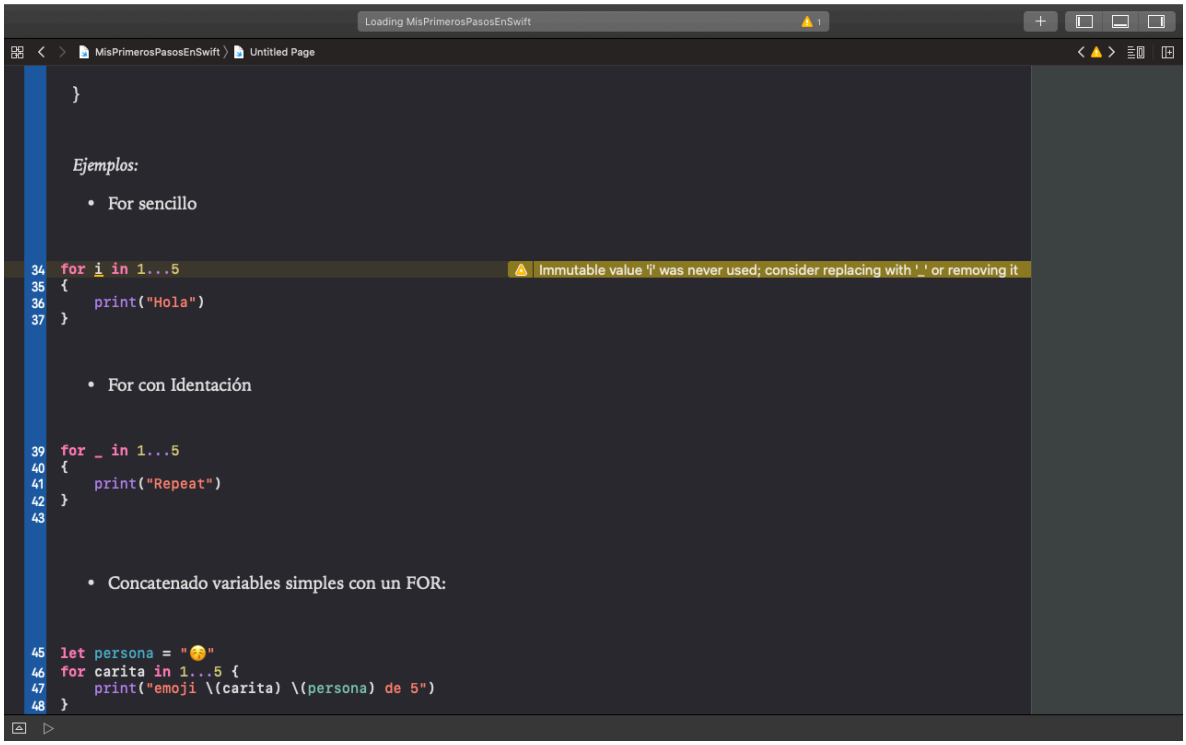

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Arrays
let tareas = ["Aspirar", "Desempolvar", "Lavar la ropa", "Alimentar a los dragones", "Lavar trastes", "Limpiar la mesa", "Lavar el baño", "Cocinar la cena", "Sacar al perro a pasear", "Ordenar mi cuarto", "Limpiar las ventanas", "Limpiar la chimenea", "Tender la ropa mojada", "Guardar la ropa limpia", "Arreglar el cajón de la cocina"]
Cada tarea te lleva realizarla alrededor de 13.454 minutos:
let minutosDeTareas = 13.454
¿Cómo averiguas cuánto tiempo te llevará aproximadamente realizarlas todas? Debes saber cuántas tareas hay en la lista. Puedes utilizar una propiedad de arrays para resolver el problema.
//Programa aquí tu solución
Anterior | Página 15 de 21 | Siguiente

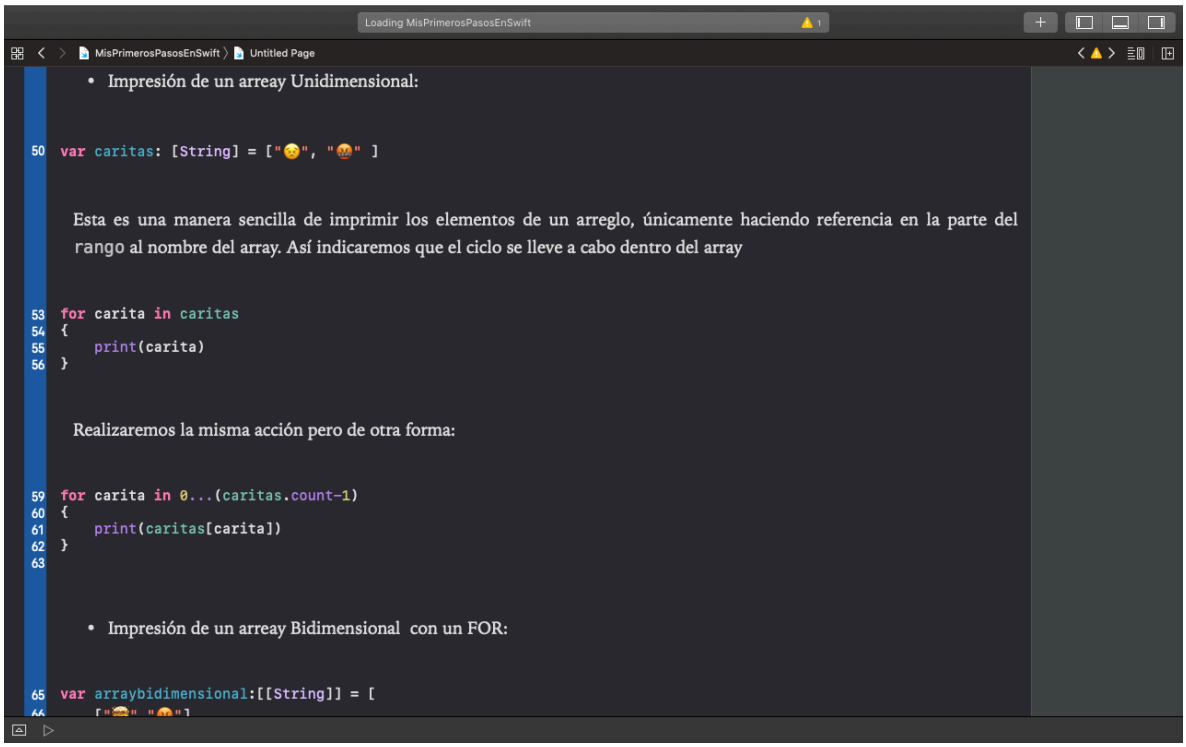

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Set y Diccionarios
✨Set
Conjunto de colecciones no ordenadas de valores únicos. Un Set almacena valores distintos de un mismo tipo en una colección sin orden definido, es decir un Set de tipo String solamente almacenará cadenas de texto y donde no podrán repetirse los valores. Usualmente se hace uso de un Set en lugar de un Array cuando el orden de los elementos no sea importante, o cuando es necesario asegurarse de que un artículo sólo aparece una sola vez.
Sintaxis:
var nombre_set : Set<Tipo_dato> = [valor1, valor2, valor3]
Ejemplo:
13 var escuelas : Set<String> = ["UNAM", "IPN", "UAM", "TEC DE MONTERREY"]
14 print(escuelas)
Utilizamos la propiedad .contains -> Para identificar si la contiene
17 if (escuelas.contains ("UNAM")) {
18 print("Tenemos esa escuela")
19 }
Podemos crear un Set vacío de un cierto tipo de dato utilizando la sintaxis de inicialización:

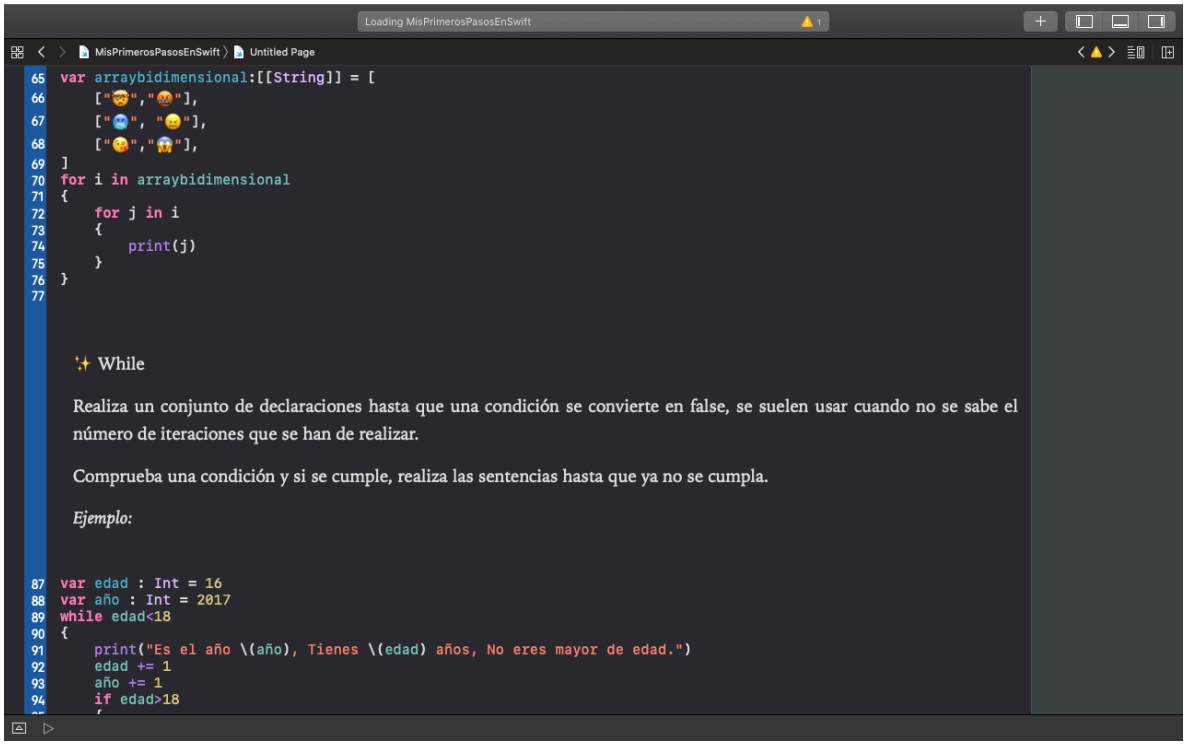

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Set y Diccionarios
24 var letras = Set<String>()
26 print("letras es de tipo Set con \(letras.count) elementos.")
✨ Diccionario
Diccionarios de colecciones no ordenados con asociaciones llave-valor. Cada valor está asociado con una clave única, que actúa como un identificador para ese valor dentro del diccionario.
A diferencia de los elementos de un Array, los elementos de un diccionario no tienen un orden específico. Hacemos uso de un diccionario cuando se necesita buscar valores en función de su identificador (llave-valor), casi de la misma manera que un diccionario en el mundo real cuando buscamos la definición de una palabra en particular.
Sintaxis:
var nombre_dicc : [Tipo_dato : [Tipo_dato]] = [llave: valor1, llave2: valor2, llave3: valor3]
Ejemplo:
41 var zoo: [String:[String]] =
42 ["Mamiferos":["🐮","🐷", "🐵", "🐶"],
43 "Voladores": ["🦅","🐞", "🐒", "🐰"],
44 "Marinos":["🐡 ", "🐙 ", "🐠"],
45 "Extintos":["🦎", "🦕"]
46]
47 print(zoo)

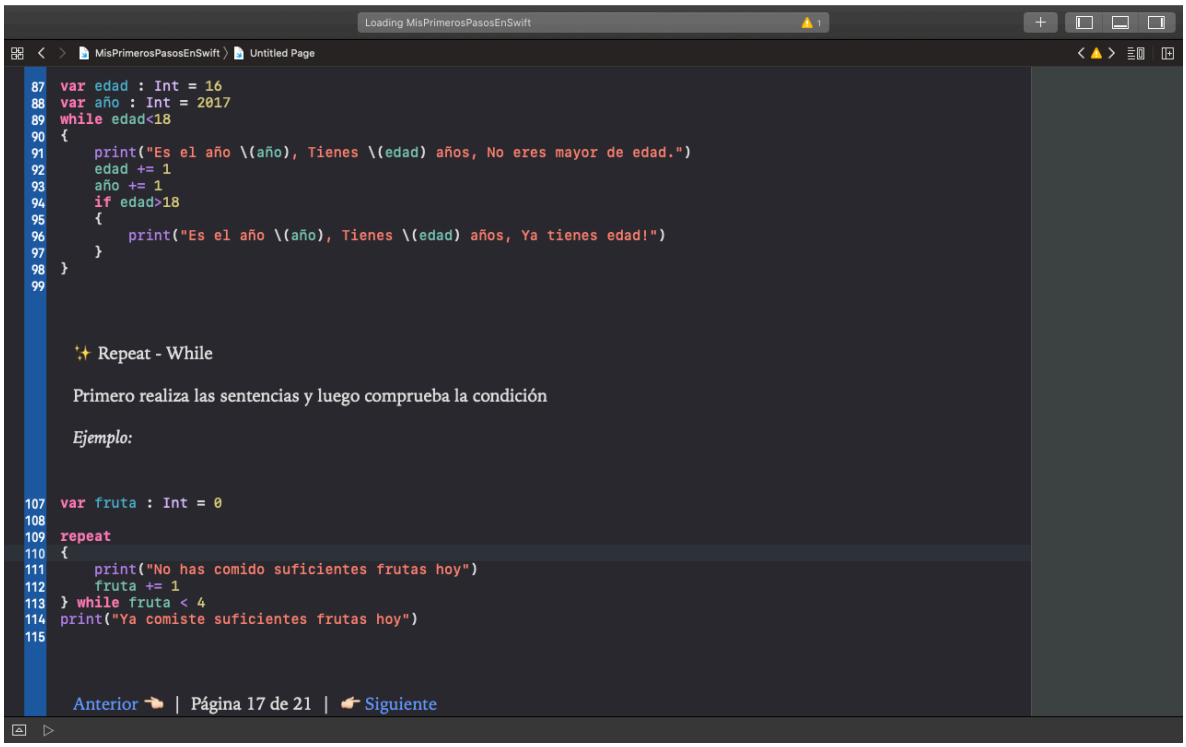

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Set y Diccionarios
Para definir un diccionario vacío, se hace de la siguiente manera:
var familia : [String : [String]] = [:]
print(familia)
Igual que en los arrays, los set y diccionarios tienen propiedades similares a las del Array, como count: Ejemplo:
print(familia.count)
Experiment
Crea un diccionario del tipo de dato que tu desees, tanto llaves como contenido. Cada llave será un genero músical y tenga dentro 3 canciones representativas del género. (Debe tener minímo 5 llaves) Investigar las siguientes propiedades:
¿Cómo imprimir únicamente las llaves de un diccionario?
¿Cómo imprimir únicamente los datos de un diccionario?
¿Cómo imprimir ambas (llave y datos)?
Anterior | Página 16 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Untitled Page
Bucles
Son bloques de código que nos facilitan como programadores a tareas que son repetitivas, ejecutando un segmento de código repetidas veces, por un número específico de iteraciones o hasta cierta condición.
Antes de comenzar con los controles de flujo recordemos los operadores de comparación y asignación que podemos manejar en Swift:
1 == 1 //true porque 1 es igual a 1
2 != 1 //true porque 2 no es equal a 1
2 > 1 //true porque 2 es más grande que 1
1 < 2 //true porque 1 más pequeño que 2
1 >= 1 //true porque 1 es mayor o igual que 1
2 <= 1 //falso porque 2 no es menor o igual que 1
For
Sirve para iterar sobre una secuencia como elementos de un array (matriz), rangos de números o caracteres en una string (cadena).
Su estructura es la siguiente:
for nombre_del_for in rango
{
Acción a repetir

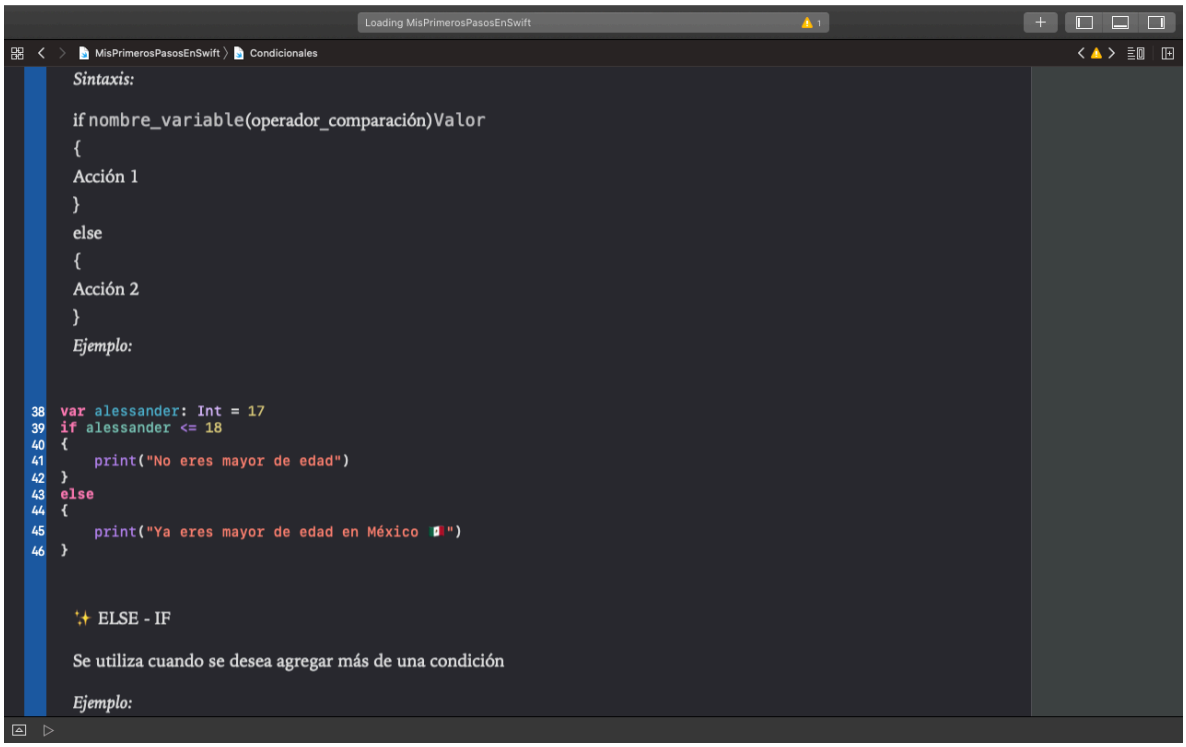

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Untitled Page
}
Ejemplos:
• For sencillo
for i in 1...5
Immutable value 'i' was never used; consider replacing with '_' or removing it
{
print("Hola")
}
• For con Identación
for _ in 1...5
{
print("Repeat")
}
• Concatenado variables simples con un FOR:
let persona = "😡"
for carita in 1...5 {
print("emoji \(carita) \(persona) de 5")
}

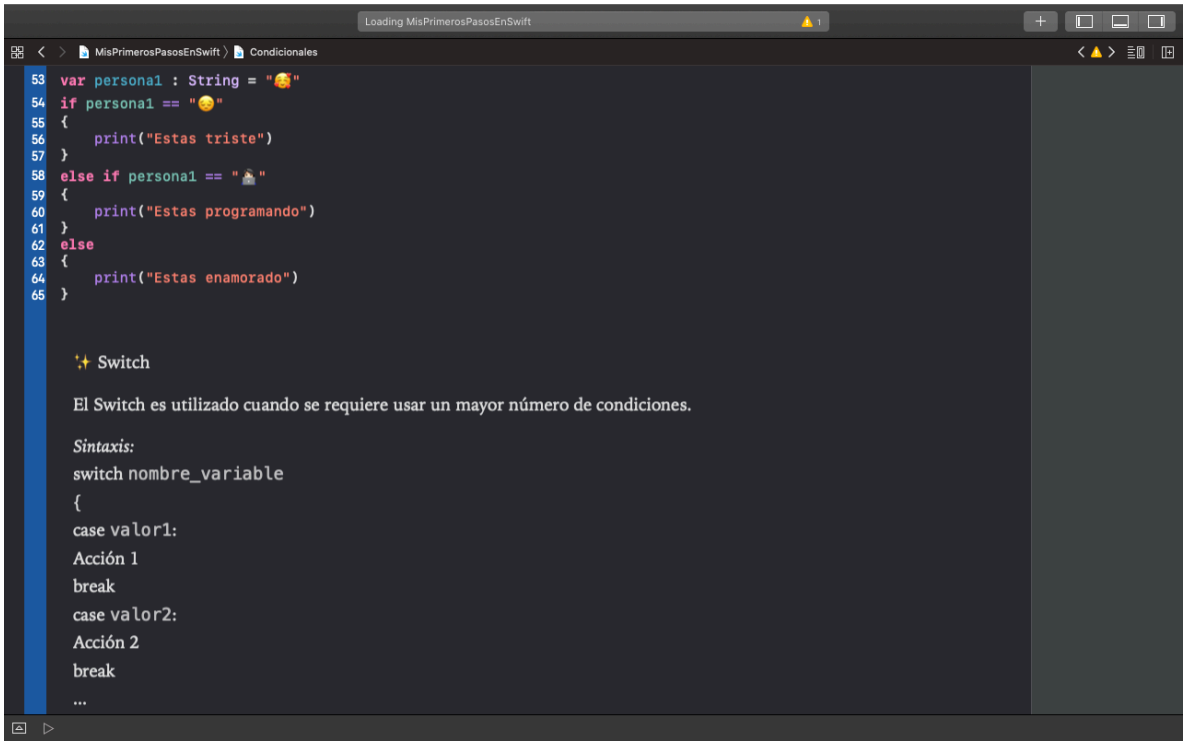

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Untitled Page
• Impresión de un arreay Unidimensional:
var caritas: [String] = ["😢", "😡"]
Esta es una manera sencilla de imprimir los elementos de un arreglo, únicamente haciendo referencia en la parte del rango al nombre del array. Así indicaremos que el ciclo se lleve a cabo dentro del array
for carita in caritas
{
print(carita)
}
Realizaremos la misma acción pero de otra forma:
for carita in 0...(caritas.count-1)
{
print(caritas[carita])
}
• Impresión de un arreay Bidimensional con un FOR:
var arraybidimensional:[[String]] = [

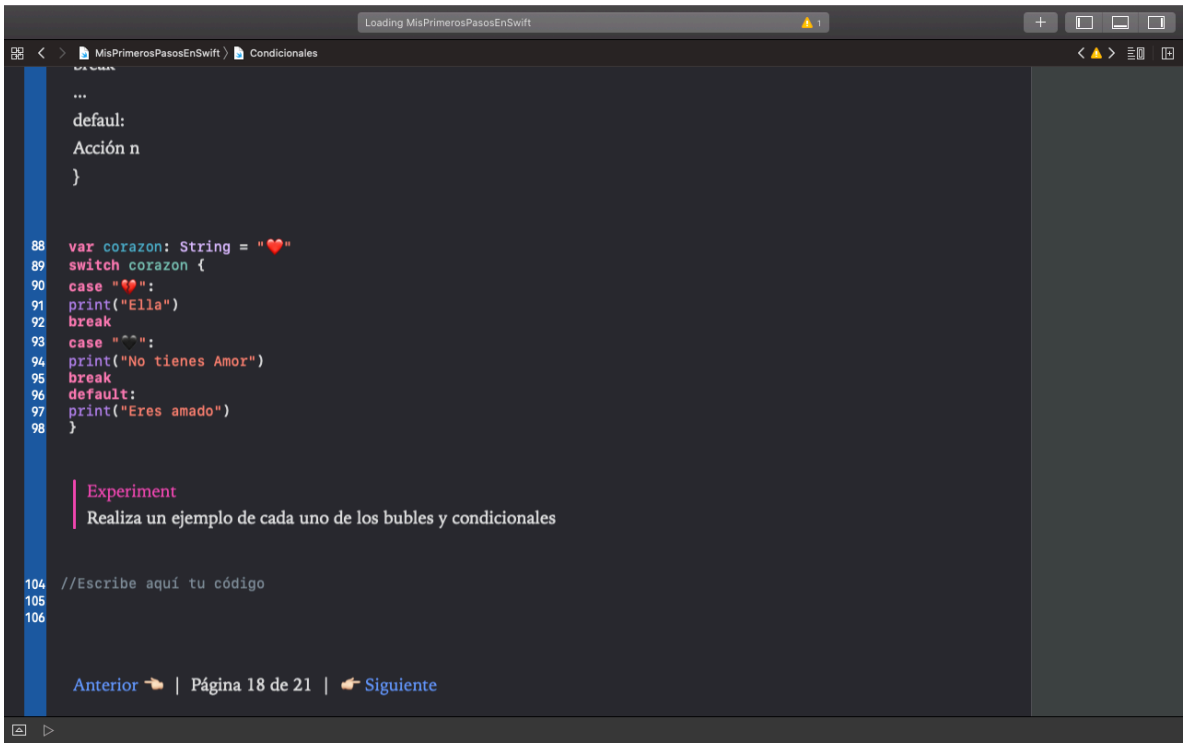

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Untitled Page
var arraybidimensional:[[String]] = [
["😍","🤠"],
["😭", "😠"],
["😘","😱"],
]
for i in arraybidimensional
{
for j in i
{
print(j)
}
}
✨ While
Realiza un conjunto de declaraciones hasta que una condición se convierte en false, se suelen usar cuando no se sabe el número de iteraciones que se han de realizar.
Comprueba una condición y si se cumple, realiza las sentencias hasta que ya no se cumpla.
Ejemplo:
var edad : Int = 16
var año : Int = 2017
while edad<18
{
print("Es el año \(año), Tienes \(edad) años, No eres mayor de edad.")
edad += 1
año += 1
if edad>18

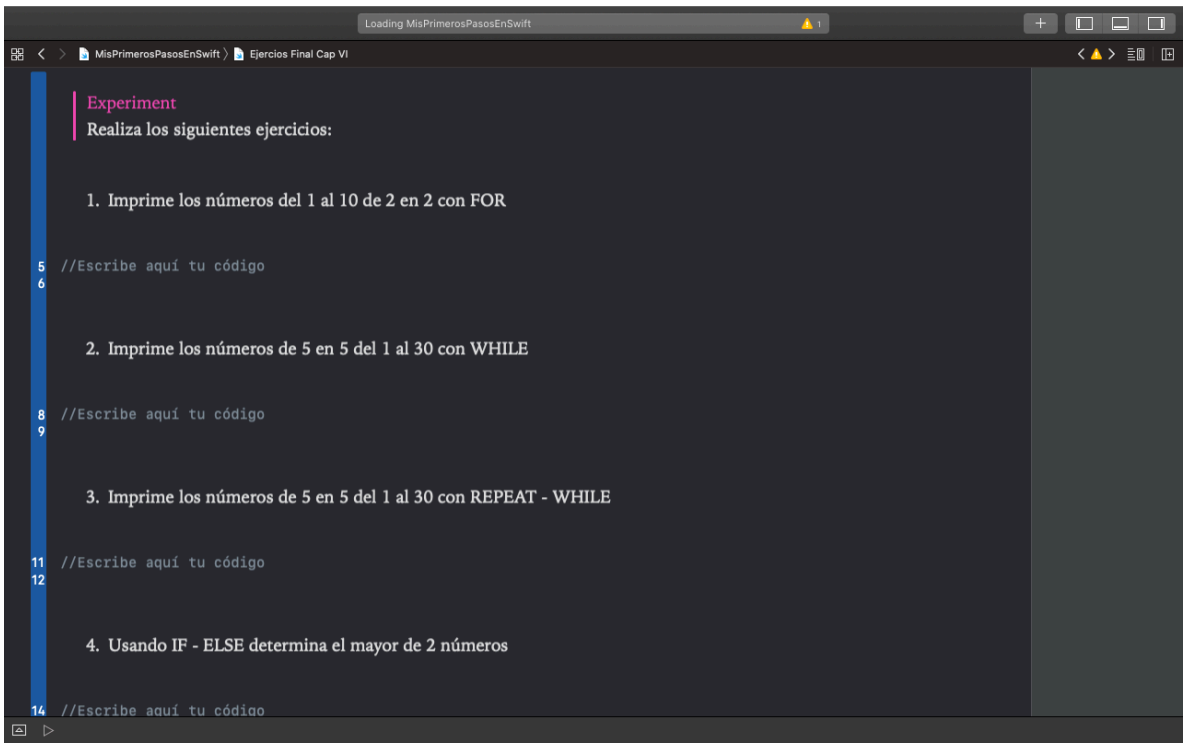

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Untitled Page
var edad : Int = 16
var año : Int = 2017
while edad<18
{
print("Es el año \(año), Tienes \(edad) años, No eres mayor de edad.")
edad += 1
año += 1
if edad>18
{
print("Es el año \(año), Tienes \(edad) años, Ya tienes edad!")
}
}
✨ Repeat - While
Primero realiza las sentencias y luego comprueba la condición
Ejemplo:
var fruta : Int = 0
repeat
{
print("No has comido suficientes frutas hoy")
fruta += 1
} while fruta < 4
print("Ya comiste suficientes frutas hoy")
Anterior | Página 17 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Condicionales
⭐ Condicionales
Las condiciones nos sirven para realizar diferentes acciones dependiendo el caso en el que nos encontremos situados. En Swift tenemos: IF-ELSE Y ELSE-IF.
✨ IF
Cuando necesitamos que una condicion se cumpla, utlizamos la sentencia If.
Sintaxis:
if nombre_variable(operador_comparación)Valor
{
Acción 1
} Ejemplo
var temperaturaFahrenheit = 30
if temperaturaFahrenheit <= 32 {
print("¡Hace mucho frío! Es mejor usar ropa abrigadadora.")
}
✨ ELSE
Si se cumple una condición, se realiza una primera acción, sino se realizar por default la segunda acción dentro del else.

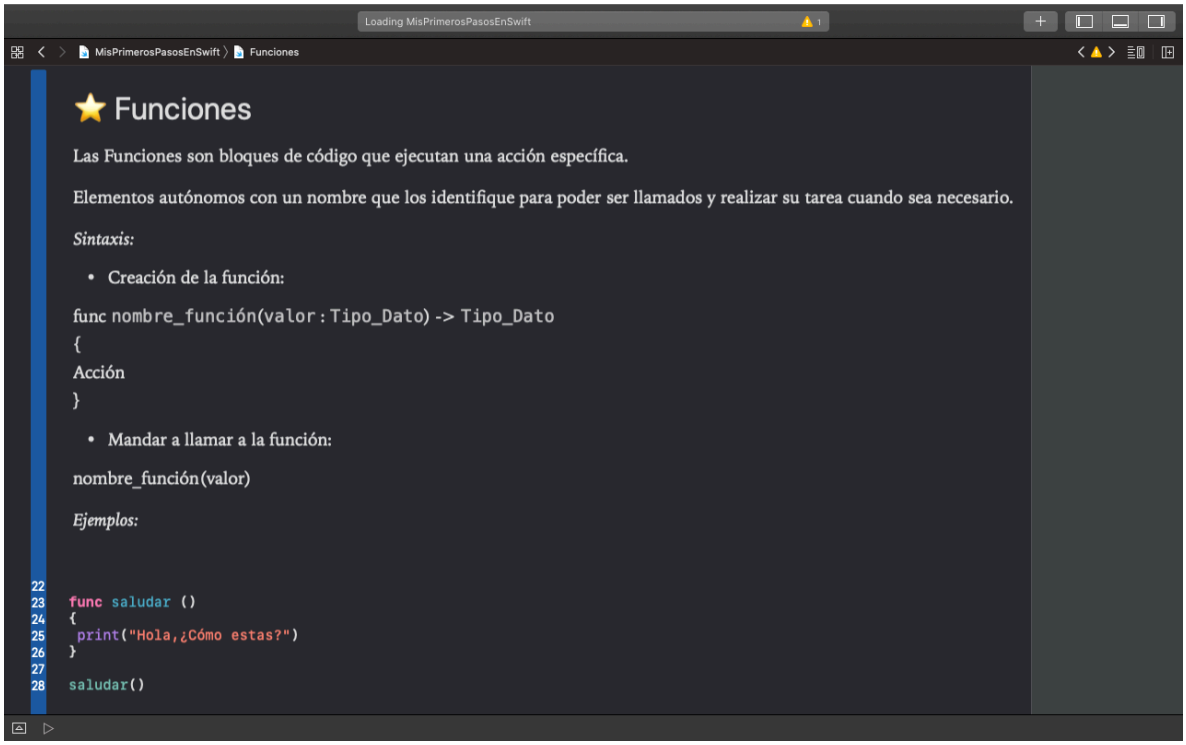

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Condicionales
Sintaxis:
if nombre_variable(operador_comparación)Valor
{
Acción 1
}
else
{
Acción 2
}
Ejemplo:
var alessander: Int = 17
if alessander <= 18
{
print("No eres mayor de edad")
}
else
{
print("Ya eres mayor de edad en México 🇲🇽")
}
✨ ELSE - IF
Se utiliza cuando se desea agregar más de una condición
Ejemplo:

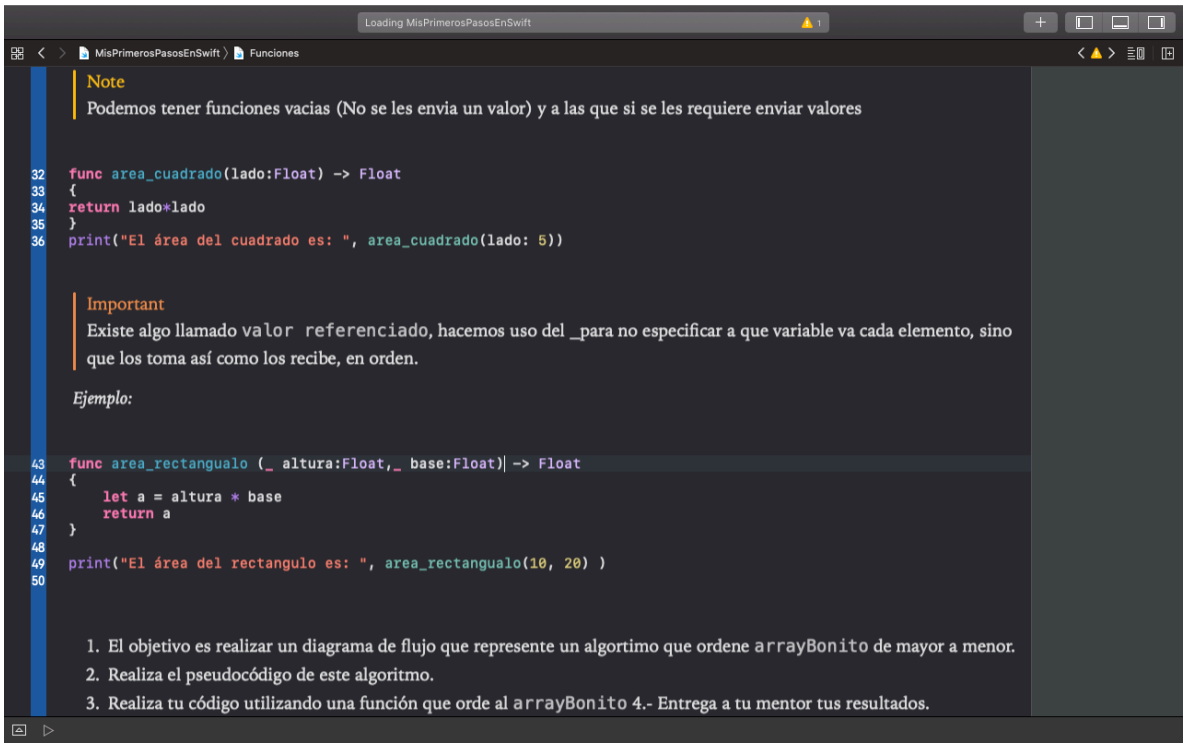

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Condicionales
var persona1 : String = "🥰"
if persona1 == "😔"
{
print("Estas triste")
}
else if persona1 == "👨🏻‍💻"
{
print("Estas programando")
}
else
{
print("Estas enamorado")
}
✨ Switch
El Switch es utilizado cuando se requiere usar un mayor número de condiciones.
Sintaxis:
switch nombre_variable
{
case valor1:
Acción 1
break
case valor2:
Acción 2
break
...


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Condicionales
...
defaul:
Acción n
}
var corazon: String = "❤️"
switch corazon {
case "💔":
print("Ella")
break
case "^^":
print("No tienes Amor")
break
default:
print("Eres amado")
}
Experiment
Realiza un ejemplo de cada uno de los bubles y condicionales
//Escribe aquí tu código
Anterior | Página 18 de 21 | Siguiente

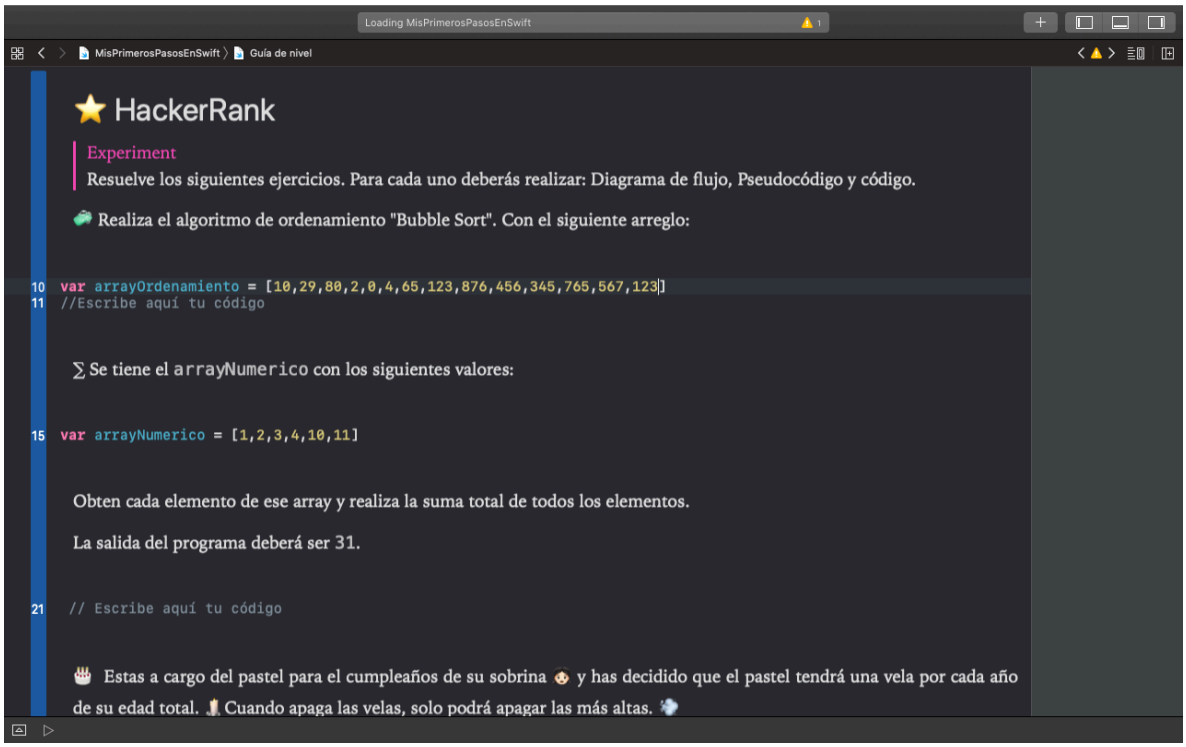

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Ejercios Final Cap VI
Experiment
Realiza los siguientes ejercicios:
1. Imprime los números del 1 al 10 de 2 en 2 con FOR
//Escribe aquí tu código
2. Imprime los números de 5 en 5 del 1 al 30 con WHILE
//Escribe aquí tu código
3. Imprime los números de 5 en 5 del 1 al 30 con REPEAT - WHILE
//Escribe aquí tu código
4. Usando IF - ELSE determina el mayor de 2 números
//Escribe aquí tu código

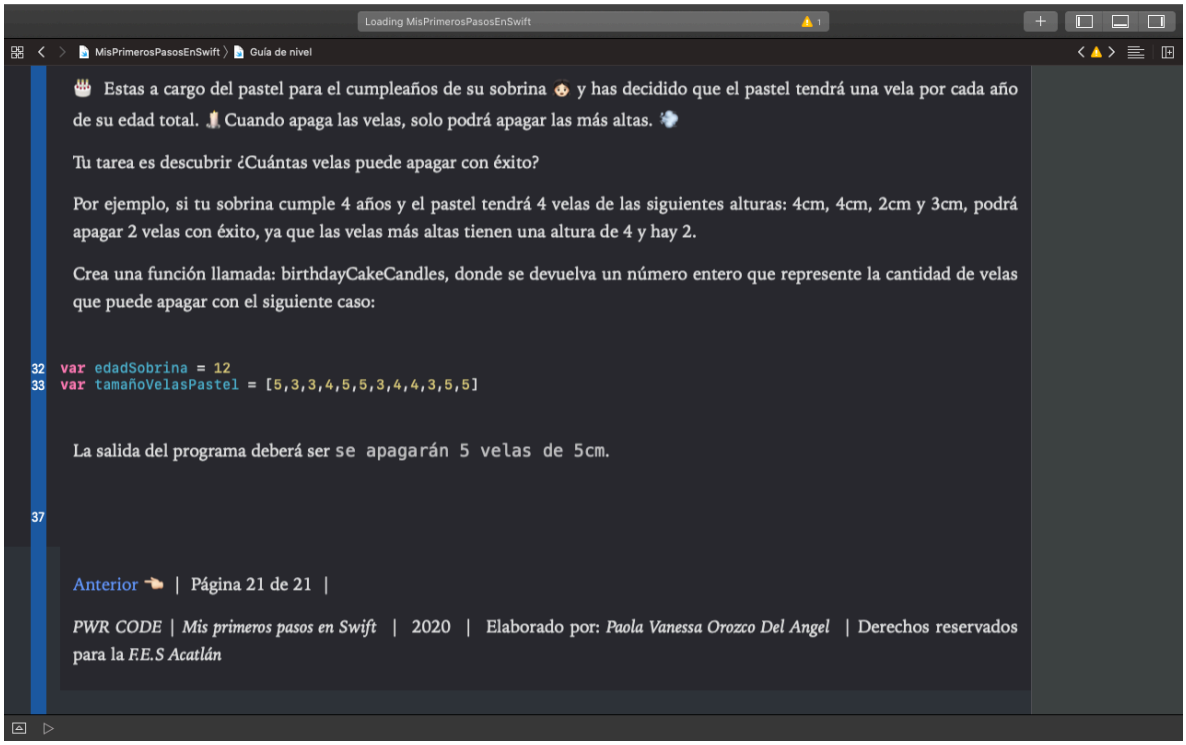

Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Ejercios Final Cap VI
//Escribe aquí tu código
4. Usando IF - ELSE determina el mayor de 2 números
//Escribe aquí tu código
5. Usando SWITCH escribe el siguiente ejercicio. Un niña va a una fiesta de cumpleaños de uno de sus amigos. Ella tiene las siguientes opciones para regalarle: , , , , , , y . Haz un programa que regrese que todos los demás regalos no haran feliz a su amigo pero los siguientes productos sí lo harán feliz: y .
//Escribe aquí tu código
Anterior | Página 19 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Funciones
⭐ Funciones
Las Funciones son bloques de código que ejecutan una acción específica.
Elementos autónomos con un nombre que los identifique para poder ser llamados y realizar su tarea cuando sea necesario.
Sintaxis:
• Creación de la función:
func nombre_función(valor : Tipo_Dato) -> Tipo_Dato
{
Acción
}
• Mandar a llamar a la función:
nombre_función(valor)
Ejemplos:
22
23 func saludar ()
24 {
25 print("Hola,¿Cómo estas?")
26 }
27
28 saludar()


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Funciones
Note
Podemos tener funciones vacias (No se les envia un valor) y a las que si se les requiere enviar valores
32 func area_cuadrado(lado:Float) -> Float
33 {
34 return lado*lado
35 }
36 print("El área del cuadrado es: ", area_cuadrado(lado: 5))
Important
Existe algo llamado valor referenciado, hacemos uso del _para no especificar a que variable va cada elemento, sino que los toma así como los recibe, en orden.
Ejemplo:
43 func area_rectangualo (_ altura:Float,_ base:Float) -> Float
44 {
45 let a = altura * base
46 return a
47 }
48
49 print("El área del rectangulo es: ", area_rectangualo(10, 20))
50
1. El objetivo es realizar un diagrama de flujo que represente un algortimo que ordene arrayBonito de mayor a menor.
2. Realiza el pseudocódigo de este algoritmo.
3. Realiza tu código utilizando una función que orde al arrayBonito 4.- Entrega a tu mentor tus resultados.


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Funciones
que los toma así como los recibe, en orden.
Ejemplo:
func area_rectangualo (_ altura:Float,_ base:Float) -> Float
{
let a = altura * base
return a
}
print("El área del rectangulo es: ", area_rectangualo(10, 20))
1. El objetivo es realizar un diagrama de flujo que represente un algortimo que ordene arrayBonito de mayor a menor.
2. Realiza el pseudocódigo de este algoritmo.
3. Realiza tu código utilizando una función que orde al arrayBonito 4.- Entrega a tu mentor tus resultados.
var arrayBonito = [3423, 5454, 6676, 12, 35, 1000, 1.67, -85]
//Escribe aquí tu código
Anterior | Página 20 de 21 | Siguiente


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Guía de nivel
HackerRank
Experiment
Resuelve los siguientes ejercicios. Para cada uno deberás realizar: Diagrama de flujo, Pseudocódigo y código.
Realiza el algoritmo de ordenamiento "Bubble Sort". Con el siguiente arreglo:
var arrayOrdenamiento = [10,29,80,2,0,4,65,123,876,456,345,765,567,123]
//Escribe aquí tu código
∑ Se tiene el arrayNumerico con los siguientes valores:
var arrayNumerico = [1,2,3,4,10,11]
Obten cada elemento de ese array y realiza la suma total de todos los elementos.
La salida del programa deberá ser 31.
// Escribe aquí tu código
Estas a cargo del pastel para el cumpleaños de su sobrina y has decidido que el pastel tendrá una vela por cada año de su edad total. Cuando apaga las velas, solo podrá apagar las más altas.


Loading MisPrimerosPasosEnSwift
MisPrimerosPasosEnSwift
Guía de nivel
Estas a cargo del pastel para el cumpleaños de su sobrina y has decidido que el pastel tendrá una vela por cada año de su edad total. Cuando apaga las velas, solo podrá apagar las más altas.
Tu tarea es descubrir ¿Cuántas velas puede apagar con éxito?
Por ejemplo, si tu sobrina cumple 4 años y el pastel tendrá 4 velas de las siguientes alturas: 4cm, 4cm, 2cm y 3cm, podrá apagar 2 velas con éxito, ya que las velas más altas tienen una altura de 4 y hay 2.
Crea una función llamada: birthdayCakeCandles, donde se devuelva un número entero que represente la cantidad de velas que puede apagar con el siguiente caso:
32 var edadSobrina = 12
33 var tamañoVelasPastel = [5,3,3,4,5,5,3,4,4,3,5,5]
La salida del programa deberá ser se apagarán 5 velas de 5cm.
37
Anterior | Página 21 de 21 |
PWR CODE | Mis primeros pasos en Swift | 2020 | Elaborado por: Paola Vanessa Orozco Del Angel | Derechos reservados para la F.E.S Acatlán