

UNIVERSIDAD POLITÉCNICA DE PACHUCA



Codeland-oop

M. EN C. ALICIA ORIZ MONTES

PROYECTO DE INVESTIGACIÓN

INTEGRANTES:

JUAN ANTONIO AYOLA CORTES

JUAN DANIEL SOTO DIMAS

MIGUEL ÁNGEL VITE HERNÁNDEZ

INGENIERÍA EN SOFTWARE

Índice general

| | |
|---|-----------|
| 1. Introducción | 1 |
| 1.1. Problemática | 1 |
| 1.1.1. Ingeniería es el área con mayor tasa de abandono escolar universitario, particularmente aquellas vinculadas con la informática | 1 |
| 1.1.2. Problemas del aprendizaje de la programación orientada a objetos . . . | 1 |
| 1.2. Objetivo general | 2 |
| 1.3. Objetivos específicos | 2 |
| 1.4. Justificación | 2 |
| 1.4.1. Las TIC en la educación | 2 |
| 1.4.2. M-Learning en la educación | 2 |
| 1.4.3. La importancia de la programación orientada a objetos | 3 |
| 2. Estado del arte | 4 |
| 3. Marco teórico | 6 |
| 3.1. Programación Orientada a Objetos | 6 |
| 3.2. Programa | 6 |
| 3.3. Tipo de dato | 6 |
| 3.4. Objeto | 6 |
| 3.5. Clase | 6 |
| 3.6. Encapsulación | 7 |
| 3.7. Abstracción | 7 |
| 3.8. Objetos | 7 |
| 3.9. Polimorfismo | 7 |
| 3.10. Herencia | 7 |
| 3.11. Atributos | 7 |
| 3.12. Métodos | 7 |
| 3.13. Algoritmo | 8 |
| 3.14. Variable | 8 |
| 3.15. Constante | 8 |
| 3.16. Parámetros formales | 8 |
| 3.17. Java | 8 |
| 3.18. Android | 8 |
| 3.19. SQLite | 8 |
| 3.20. Material Design | 8 |
| 3.21. Git | 8 |
| 3.22. UML | 9 |
| 3.23. Android Studio | 9 |
| 4. Desarrollo | 10 |
| 4.1. Recursos | 10 |
| 4.1.1. Recursos humanos | 10 |
| 4.1.2. Recursos materiales | 10 |
| 4.1.3. Recursos tecnológicos | 10 |

| | |
|---|-----------|
| 4.1.4. Recursos administrativos | 11 |
| 4.2. Diagrama de Gantt | 11 |
| 4.3. Encuesta | 11 |
| 4.4. Historias de usuario | 13 |
| 4.5. Grafica de quemado | 15 |
| 4.6. Maquetado | 16 |
| 5. Resultados | 17 |
| 6. Anexos | 20 |
| 6.1. Plan de pruebas | 20 |
| 6.2. Casos de pruebas | 23 |

Índice de figuras

| | |
|---|----|
| 1.1. Evolución de las TIC en la educación. (Basado en Palacios et al. (2011)) | 3 |
| 4.1. Cronograma de actividades a seguir. | 11 |
| 4.2. Primera pregunta de la encuesta.. . . . | 11 |
| 4.3. Segunda pregunta del cuestionario (parte 1). | 12 |
| 4.4. Segunda pregunta del cuestionario (parte 2). | 12 |
| 4.5. Tercera pregunta del cuestionario. | 13 |
| 4.6. Grafica de quemado de las historias de usuario. | 15 |
| 4.7. Pantallas de las fases, temas, tema, ejercicios y ejemplo. | 16 |
| 5.1. Pantalla de inicio donde se muestran las fases. | 17 |
| 5.2. Pantalla de selección de los temas de cada fase. | 18 |

Índice de cuadros

| | |
|---|----|
| 2.1. Aplicaciones existentes para aprender programación orientada a objetos | 4 |
| 4.1. Personal requerido. | 10 |
| 4.2. Recursos materiales. | 10 |
| 4.3. Recursos tecnológicos | 10 |
| 4.4. Recursos administrativos | 11 |
| 4.5. Historia de usuario 1 | 13 |
| 4.6. Historia de usuario 2 | 13 |
| 4.7. Historia de usuario 3 | 13 |
| 4.8. Historia de usuario 4 | 14 |
| 4.9. Historia de usuario 5 | 14 |
| 4.10. Historia de usuario 6 | 14 |
| 4.11. Historia de usuario 7 | 14 |
| 6.2. Caso de prueba numero 1 | 23 |
| 6.3. Caso de prueba numero 2 | 23 |
| 6.4. Caso de prueba numero 3 | 24 |
| 6.5. Caso de prueba numero 4 | 24 |
| 6.6. Caso de prueba numero 5 | 24 |

Capítulo 1

Introducción

La programación orientada a objetos es uno paradigma de suma importancia dentro de la programación por lo que es básico conocerlo, y resulta muy útil que los estudiantes que cursan carreras técnicas o universitarias enfocadas al ámbito de la programación tengan un alto conocimiento de esto, sin embargo a pesar de llevarla como materia, les es difícil aprenderla por los amplios conceptos que maneja. Esto también conlleva a otro problema, que es la deserción de los alumnos de la carrera. La reprobación estudiantil es un problema añejo y complejo en las instituciones de educación superior. El propósito de este proyecto es aumentar el egreso de estudiantes.

1.1. Problemática

1.1.1. Ingeniería es el área con mayor tasa de abandono escolar universitario, particularmente aquellas vinculadas con la informática

El mercado laboral sufre el dilema vocacional porque las carreras más demandadas son las menos elegidas por los estudiantes después del primero año, ya que el abandono es mayor en el área de ingeniería y especialmente en informática. El foro profesional de los ingenieros técnicos en España advierte que el déficit de profesionales en el área de software y telecomunicaciones derivará en la importancia masiva de ingenieros dentro de una década.

1.1.2. Problemas del aprendizaje de la programación orientada a objetos

El problema del aprendizaje de la programación orientada a objetos se manifiesta toda vez que es una materia compleja que implica la integración de muchos elementos como son el paradigma orientado a objetos, el lenguaje de programación, el entorno de desarrollo, la metodología de desarrollo, el lenguaje de modelado, los patrones de desarrollo y la lógica de programación. Por lo tanto los alumnos se encuentran ante una cantidad abrumadora de conceptos en un periodo corto de tiempo, lo que dificulta su asimilación y el desarrollo de las habilidades para generar líneas de código como lo explica Spigariol and Passerini (2013):

“Los docentes veían en los estudiantes que el uso del lenguaje representaba una curva de aprendizaje abrupta en los primeros momentos de la materia ya que requieren el manejo de una cantidad amplia de conceptos antes de poder realizar algo relativamente sencillo (...). La disociación entre teoría y práctica que se generaba era ciertamente contraproducente y dificultaba el proceso de aprendizaje.”

Debido a que los conceptos que se tratan en la asignatura de programación orientada a objetos son muchos y en algunos casos difíciles de comprender por el alto nivel de abstracción, esto se configura como un factor que dificulta el aprendizaje, esto se ha visto reflejado por

ejemplo cuando los estudiantes generalmente no distinguen entre lo que es una clase y un objeto.

Además la forma de enseñar la asignatura de programación orientada a objetos es muy similar a la de la programación estructurada, primero se tratan temas básicos del lenguaje de programación, como son la declaraciones de los tipos de datos, las estructuras de control y las sentencias de condición, para posteriormente enseñar lo que son las clases, objetos y los principales temas propios del paradigma orientada a objetos, esto contribuye a que los estudiantes sientan que continúan con el mismo paradigma de programación estructurada.

En virtud de que los estudiantes no logran asimilar este cambio de paradigmas, usan el lenguaje de programación orientada a objetos como un lenguaje de programación estructurada, por consiguiente no resuelven problemas diseñando clases.

Así pues los estudiantes al llegar a la asignatura de programación orientada a objetos se enfrentan de entrada a dos problemas, la percepción que tienen de dificultad y por otro lado el proceso de transición del paradigma estructurado hacia el paradigma orientado a objetos.

1.2. Objetivo general

Desarrollar una aplicación móvil con la herramienta de Android Studio, para que las personas interesadas en la programación puedan aprender de una mejor manera el paradigma de programación orientada a objetos, y así puedan adquirir y/o reforzar su conocimiento de este paradigma.

1.3. Objetivos específicos

- Proporcionar a los estudiantes y personas interesadas en la programación, información acerca del paradigma de programación orientada a objetos.
- Evaluar el conocimiento de los usuarios de la aplicación con exámenes al finalizar un tema.
- Diseñar una interfaz de tal forma que el usuario pueda aprender a utilizar la aplicación en el menor tiempo posible.
- Programar la aplicación de tal forma de que no consuma muchos recursos al usarla en un celular.

1.4. Justificación

1.4.1. Las TIC en la educación

De acuerdo con Palacios et al. (2011) en la educación el uso de las nuevas tecnologías de la información ha pasado por las siguientes etapas:

Esta evolución muestra como cada vez el aprendizaje va utilizando más tecnología; pasando de ser ésta simplemente una herramienta de apoyo, hasta ser la plataforma a través de la cual se presentan los contenidos y evalúan los conocimientos.

Mix-Learning. La etapa posterior al e-learning es la aplicación de una mezcla de sus herramientas con sistemas educativos tradicionales. La finalidad es dirigir más específicamente los contenidos a los estudiantes. Es así que el Blend Learning, Mix Learning o Hybrid Learning se presenta como “la combinación efectiva de los diferentes modelos de reparto, modelos de enseñanza y modelos de aprendizaje”(Olivares Carmona et al., 2016).

1.4.2. M-Learning en la educación

Según la Organización de las Naciones Unidas para la Educación la Ciencia y la Cultura (Dykes and Knight, 2012) existen 5.9 billones de suscripciones de teléfonos móviles en el

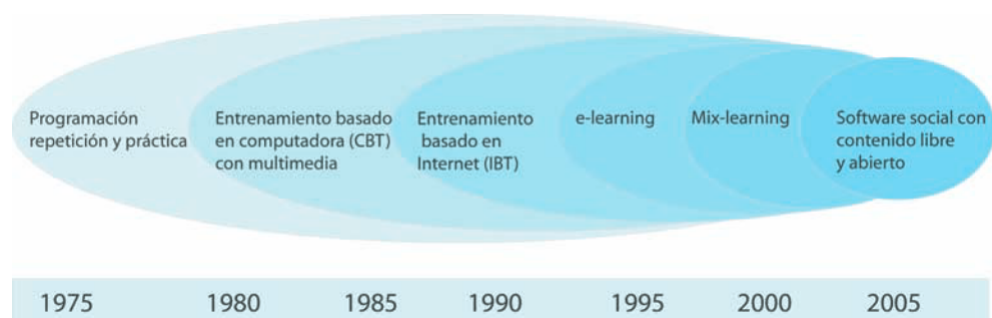


Figura 1.1: Evolución de las TIC en la educación. (Basado en Palacios et al. (2011))

mundo, contra los 7.04 billones de habitantes. Además, en el año 2020 los dispositivos móviles serán la principal herramienta de conexión a internet para la mayoría de la población; en Japón, actualmente el 75 % de su población tiene un dispositivo móvil como primer medio de acceso a internet (Camacho and Lara, 2011).

Asimismo, en años recientes el uso de la tecnología móvil para fines educativos, conocido como m-learning, ha tenido un gran desarrollo en la educación superior, ya que existen universidades de Europa y América que cuentan con sistemas de educación móvil (Traxler, 2007).

1.4.3. La importancia de la programación orientada a objetos

La programación Orientada a Objetos surge como el paradigma que permite manejar ampliamente las nuevas plataformas que garanticen desarrollar aplicaciones robustas, portables y reutilizables que puedan ofrecer una solución a largo plazo en un mundo donde los cambios se dan a cada momento.

El desarrollo de programas orientados a objetos es un enfoque diferente del mundo informático. Implica la creación de modelos del mundo real y la construcción de programas informáticos basados en esos modelos.

La importancia de esta programación radica en que, favorece la creación de programas de calidad, fuerza en mantenimiento, en extensión y reutilización de programas. Está basada en el modo de pensar del hombre y en el modo de trabajar de la máquina.

Es muy importante que los estudiantes sean capaces, no sólo de manejar los conceptos de orientación a objetos, sino también de aplicarlos de manera efectiva en el desarrollo de programas.

Capítulo 2

Estado del arte

Cuadro 2.1: Aplicaciones existentes para aprender programación orientada a objetos

| Nombre | Temas | Juegos | Tamaño | Idioma | Test |
|----------------------------------|---|--------|--------|---------|------|
| Programación Orientada a Objetos | <ul style="list-style-type: none">-Tipo de dato anónimo-Tipo abstracto-SOLID-RAII-Proxy-Programación basada en prototipos-Problema del diamante-Principio de sustitución de Liskov-Principio de segregación de la interfaz-Principio de responsabilidad única-Objeto-Método-Metaclass-Encapsulamiento-Destructor-Delegación-Clase-Campo | No | 3.87M | Español | No |
| Object Oriented Programming | <ul style="list-style-type: none">-Object-Class-Constructor-Destructor-Get-Set-toString-Private Method-Protected Method-Public Method-Inheritance-Interfaces-Abstract-Polymorphism-Encapsulation | No | 9.64M | Ingles | No |

Table 2.1 continued from previous page

| | | | | | |
|--------------------------------|--|----|-------|--------|----|
| Object Oriented Programming | <ul style="list-style-type: none"> -OOP in PHP - What is Object -OOP in PHP - Intro to Class -OOP in PHP - Inheritance -OOP in PHP - Interfaces -OOP in PHP - Abstract -OOP in PHP - Constructor -OOP in PHP - Polymorphism -OOP in PHP - Encapsulation -OOP in PHP - Destructor -OOP in PHP - Private Method -OOP in PHP - Protected Method -OOP in PHP - Public Method | No | 9.71M | Ingles | No |
| Object Oriented Programming | <ul style="list-style-type: none"> -Introduction to Object. -Class in OOP and its detail with syntax code. -Abstraction concepts in OOP -What is Polymorphism in OOP -Inheritance concepts with syntax code. -Method overloading vs. overriding -Encapsulation concept -Keywords in java -Constructor in Java etc. . . -Java Programming Statements with code syntax. | No | 2.8M | Ingles | No |
| OPP for Beginners | <ul style="list-style-type: none"> -Naming convention -Object and class -Method overloading -Constructor -Static keyword -Inheritance -Aggregation -Method overrridong -covariant return type -Abstract class -Package -object class | No | 8.1M | Ingles | Si |

Capítulo 3

Marco teórico

3.1. Programación Orientada a Objetos

La programación orientada a objetos, ha tomado las mejores ideas de la programación estructurada y los ha combinado con varios conceptos nuevos y potentes que incitan a contemplar las tareas de programación desde un nuevo punto de vista. La programación orientada a objetos, permite descomponer más fácilmente un problema en subgrupos de partes relacionadas del problema. Entonces, utilizando el lenguaje se pueden traducir estos subgrupos a unidades auto contenidas llamadas objetos.. (Izquierdo, 2007).

3.2. Programa

Un programa es una secuencia lógica de instrucciones escritas en un determinado lenguaje de programación que dicta a la computadora las acciones que debe llevar a cabo.

3.3. Tipo de dato

Es la representación simbólica de un atributo de una entidad; en programación, los datos expresan características de las entidades sobre las que opera un algoritmo. Los datos representan hechos, observaciones, cantidades o sucesos y pueden tomar la forma de números, letras o caracteres especiales (Izquierdo, 2007).

3.4. Objeto

Una estructura de datos y conjunto de procedimientos que operan sobre dicha estructura. Una definición más completa de objeto es: una entidad de programa que consiste en datos y todos aquellos procedimientos que pueden manipular aquellos datos; el acceso a los datos de un objeto es solamente a través de estos procedimientos, únicamente estos procedimientos pueden manipular, referenciar y/o modificar estos datos.

Para poder describir todos los objetos de un programa, conviene agrupar éstos en clases (Izquierdo, 2007).

3.5. Clase

Podemos considerar una clase como una colección de objetos que poseen características y operaciones comunes. Una clase contiene toda la información necesaria para crear nuevos objetos (Izquierdo, 2007).

3.6. Encapsulación

Es una técnica que permite localizar y ocultar los detalles de un objeto. La encapsulación previene que un objeto sea manipulado por operaciones distintas de las definidas. La encapsulación es como una caja negra que esconde los datos y solamente permite acceder a ellos de forma controlada (Izquierdo, 2007).

3.7. Abstracción

En el sentido más general, una abstracción es una representación concisa de una idea o de un objeto complicado. En un sentido más específico, la abstracción localiza y oculta los detalles de un modelo o diseño para generar y manipular objetos (Izquierdo, 2007).

3.8. Objetos

Un objeto es una entidad lógica que contiene datos y un código que manipula estos datos; el enlazado de código y de datos, de esta manera suele denominarse encapsulación. Cuando se define un objeto, se está creando implícitamente un nuevo tipo de datos (Izquierdo, 2007).

3.9. Polimorfismo

Esta característica es la capacidad que objetos similares tienen para responder de diferentes formas al mismo mensaje, y permite al programador implementar múltiples formas de un mismo método, dependiendo cada una de ellas de la clase sobre la que se realice la implementación. Esto permite acceder a varios métodos distintos utilizando el mismo medio de acceso (el mismo nombre). El polimorfismo está muy relacionado con la herencia (Velarde de Barraza et al., 2006).

3.10. Herencia

Según Joyanes Aguilar (1996), la herencia “Es la capacidad para crear nuevas clases de objetos que se construyen basados en clases existentes”. La herencia es una propiedad que permite a un objeto poseer propiedades de otras clases. Además, a estos nuevos objetos creados es posible asignarles nuevos atributos y métodos.

La clase que puede ser heredada se denomina clase base (superclase) y la clase que hereda se denomina clase derivada (subclase) (Velarde de Barraza et al., 2006).

3.11. Atributos

Describen las características de los objetos: tipo de acceso (privado, protegido, público) y tipo de dato (entero, real, booleano, etcétera) (Izquierdo, 2007).

3.12. Métodos

Describen lo que puede hacer la clase; es decir, el método define las instrucciones necesarias para realizar un proceso o tarea específicos. La definición del método se compone de tipo de acceso, tipo de retorno, nombre del método, parámetros, si los requiere, y el cuerpo del método (Izquierdo, 2007).

3.13. Algoritmo

Se define como una técnica de solución de problemas que consiste en una serie de instrucciones paso por paso y que produce resultados específicos para un problema determinado (Izquierdo, 2007).

3.14. Variable

Es un área de almacenamiento temporal a la que se ha asignado un nombre simbólico y cuyo valor puede ser modificado a lo largo de la ejecución de un programa (Izquierdo, 2007).

3.15. Constante

Es un valor definido que no cambia durante la ejecución de un programa (Izquierdo, 2007).

3.16. Parámetros formales

Son variables que reciben valores desde el punto de llamada que activa al método (Izquierdo, 2007).

3.17. Java

Java es uno de los lenguajes de programación más populares del mundo. Es un lenguaje orientado a objetos, potente, versátil y multiplataforma (corre en cualquier sistema operativo moderno). Java fue elegido como el lenguaje para el entorno de desarrollo de Android, el sistema operativo móvil líder en smartphones y tablets (Izquierdo, 2007).

3.18. Android

Android es un sistema operativo inicialmente pensado para teléfonos móviles, al igual que iOS, Symbian y Blackberry OS. Lo que lo hace diferente es que está basado en Linux, un núcleo de sistema operativo libre, gratuito y multiplataforma (Studio, 2016).

3.19. SQLite

SQLite es un sistema de gestión de bases de datos relacional compatible con ACID, contenida en una relativamente pequeña (275 kiB) biblioteca escrita en C (Owens and Allen, 2010).

3.20. Material Design

Material Design es un concepto, una filosofía, unas pautas enfocadas al diseño utilizado en Android.

3.21. Git

Git es un sistema de control de versiones distribuido cuyo objetivo es el de permitir mantener una gran cantidad de código a una gran cantidad de programadores eficientemente (Alvarez and Alcázar, 2014).

3.22. UML

UML son las siglas de “Unified Modeling Language” o “Lenguaje Unificado de Modelado”. Se trata de un estándar que se ha adoptado a nivel internacional por numerosos organismos y empresas para crear esquemas, diagramas y documentación relativa a los desarrollos de software.

3.23. Android Studio

Es el entorno de desarrollo integrado (IDE) oficial para el desarrollo de aplicaciones para Android y se basa en IntelliJIDEA (Studio, 2016).

Capítulo 4

Desarrollo

Este proyecto se desarrollará con la metodología SCRUM con 8 sprints de una revision por semana, haciendo uso de historias de usuario y de la grafica del quemado para llevar un control de que historias de usuario se quemaron.

4.1. Recursos

4.1.1. Recursos humanos

| Rol | Personas requeridas | Salario |
|-------------------|---------------------|----------|
| Programador | 1 | \$15,000 |
| Diseñador | 1 | \$10,000 |
| Tester | 1 | \$10,000 |
| Lider de proyecto | 1 | \$15,000 |

Cuadro 4.1: Pesonal requerido.

4.1.2. Recursos materiales

| Equipo | Precio | Unidades necesarias |
|-----------------|---------|---------------------|
| Computadoras | \$8,000 | 2 |
| Celular Android | \$5,000 | 1 |

Cuadro 4.2: Recursos materiales.

4.1.3. Recursos tecnológicos

| Concepto | Precio |
|----------------|--------|
| GanttProject | \$0 |
| Pencil | \$0 |
| Android Studio | \$0 |

Cuadro 4.3: Recursos tecnológicos

4.1.4. Recursos administrativos

| Concepto | Precio |
|--------------|--------------|
| Agua | \$100 al mes |
| Electricidad | \$300 al mes |
| Internet | \$500 al mes |

Cuadro 4.4: Recursos administrativos

4.2. Diagrama de Gantt

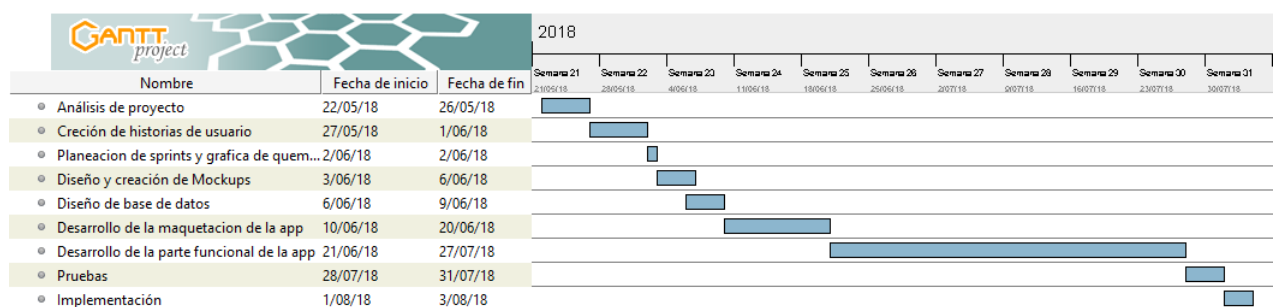


Figura 4.1: Cronograma de actividades a seguir.

4.3. Encuesta

Se realizó una encuesta a los alumnos de la carrera de Ingenieria en Software del 3° cutatrimestre de la Universidad Politecnica de Pachuca, con el fin de que con los resultados poder formular las historias de usuario. Se obtuvieron los siguientes resultados:

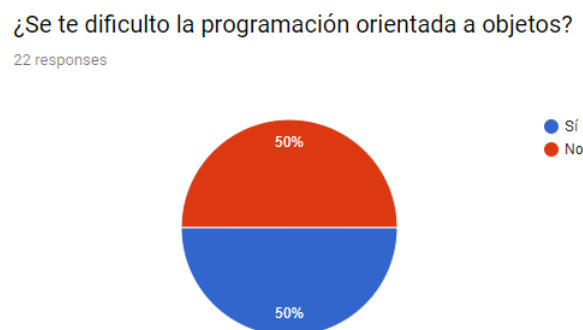


Figura 4.2: Primera pregunta de la encuesta..

En la primera pregunta se puede observar que a un 50 % de los alumnos de les dificulta la programación orientada a objetos, mientras que al 50 % restante no se le dificulta.

Si existiera una aplicación que te ayudara a aprender poo ¿Cómo te gustaría que fuera?

23 responses

Muy fácil (3)

Que tuviera información específica del tema y videos tutoriales para poder facilitar la comprensión de los temas (2)

Este mensaje es de prueba

Didáctica

De un modo didáctica y de aprendizaje mediante ejercicios.

Una aplicación en la que se puedan hacer prácticas, y que sea interactiva y incluyera ejercicios y videos

Más dinámica

Modalidad de juegos, con breves capsulas que expliquen la teoría.

Más intuitiva y más visual

Si es para aprender desde 0 que sea de una forma en la que se vean mas ejemplos y sin utilizar tantos tecnicismos.
Si es para personas que ya tienen una idea sobre la programación se puede usar un poco mas de tecnicismos a los cuales ya estén acostumbrados.

Figura 4.3: Segunda pregunta del cuestionario (parte 1).

Si existiera una aplicación que te ayudara a aprender poo ¿Cómo te gustaría que fuera?

23 responses

Si es para aprender desde 0 que sea de una forma en la que se vean mas ejemplos y sin utilizar tantos tecnicismos.
Si es para personas que ya tienen una idea sobre la programación se puede usar un poco mas de tecnicismos a los cuales ya estén acostumbrados.

Bastante didáctica y con muchos ejemplos para aplicar lo aprendido.

Que explicara a detalle cómo hacer las cosas y para que podrían servir

Visual y práctica

Práctica y con videos de ayuda

Gratuita y entendible

Didáctica

Intuitiva y con muchos ejercicios

Intuitiva y práctica

Muy explícita

Que interpretara el código y lo compilara a la vez, robusta y con indiferencia a la arquitectura.

Figura 4.4: Segunda pregunta del cuestionario (parte 2).

En la segunda pregunta, tres alumnos respondieron que la aplicación fuera "muy fácil," mientras que dos alumnos respondieron que la aplicación contuviera información muy específica del tema y tutoriales para facilitar la comprensión del tema, mientras que los demás alumnos respondieron con respuestas como que sea didáctica, dinámica, intuitiva, con una modalidad de juegos, que no contenga muchos tecnicismos y con videos de ayuda.

¿Usarías una aplicación así?

22 responses

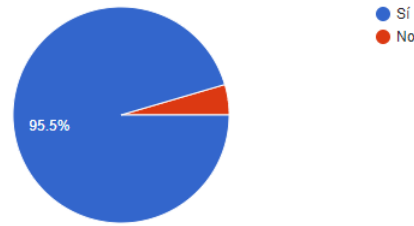


Figura 4.5: Tercera pregunta del cuestionario.

En la tercera pregunta del cuestionario, el 95.5 % de los alumnos repondieron afirmativamente a que si utilizarían una aplicación de acuerdo a lo que respondieron anteriormente, mientras que el 4.5 % restante respondió de forma negativa.

4.4. Historias de usuario

| 1. Aplicación didáctica | | |
|--|-------------------|---------------|
| Como usuario quisiera que la aplicación fuera de un modo didáctica y de aprendizaje mediante ejercicios. | | |
| Estimación: 7 | semanas Valor: 50 | Dependencias: |
| Condiciones de satisfacción: <ul style="list-style-type: none"> La aplicación contendrá información selecta y bien explicada. Se incluirán algunos ejercicios los cuales deberán resolverse en la computadora. | | |

Cuadro 4.5: Historia de usuario 1

| 2. Explicaciones detalladas | | |
|--|-------------------|---------------|
| Como usuario quisiera que explicara a detalle cómo hacer las cosas y para que podrían servir. | | |
| Estimación: 7 | semanas Valor: 60 | Dependencias: |
| Condiciones de satisfacción: <ul style="list-style-type: none"> Se dará una aplicación detallada de cada tema así como ejemplos de uso. | | |

Cuadro 4.6: Historia de usuario 2

| 3. Modalidad de juegos | | |
|--|-------------------|---------------|
| Como usuario quisiera que la aplicación tuviera la modalidad de juegos, con breves capsulas que expliquen la teoría. | | |
| Estimación: 6 | semanas Valor: 60 | Dependencias: |
| Condiciones de satisfacción: <ul style="list-style-type: none"> Se pondrá información sobre algunos conceptos y se hará un pequeño test de manera que ayude a recordar. | | |

Cuadro 4.7: Historia de usuario 3

| 4. Información específica | | |
|---|-------------------|---------------|
| Como usuario quisiera que tuviera información específica del tema y videos tutoriales para poder facilitar la comprensión de los temas. | | |
| Estimación: 6 | semanas Valor: 30 | Dependencias: |
| Condiciones de satisfacción: <ul style="list-style-type: none"> Se dará la información de manera específica pero en lugar de videos se incluirán algunos ejemplos. | | |

Cuadro 4.8: Historia de usuario 4

| 5. Visual y practica | | |
|--|-------------------|---------------|
| Como usuario quisiera que la aplicación fuera visual y práctica. | | |
| Estimación: 4 | semanas Valor: 20 | Dependencias: |
| Condiciones de satisfacción: <ul style="list-style-type: none"> La aplicación será visual y de fácil utilización. | | |

Cuadro 4.9: Historia de usuario 5

| 6. Intuitiva | | |
|--|-------------------|---------------|
| Como usuario quisiera que la aplicación fuera intuitiva y con muchos ejercicios. | | |
| Estimación: 3 | semanas Valor: 20 | Dependencias: |
| Condiciones de satisfacción: <ul style="list-style-type: none"> Se creará una interfaz amigable con el usuario y fácil de aprender. | | |

Cuadro 4.10: Historia de usuario 6

| 7. Con ejemplos y pocos tecnicismos | | |
|--|-------------------|---------------|
| Como usuario quisiera que si es para aprender desde 0 que sea de una forma en la que se vean más ejemplos y sin hacer uso excesivo de tecnicismos. | | |
| Estimación: 3 | semanas Valor: 20 | Dependencias: |
| Condiciones de satisfacción: <ul style="list-style-type: none"> Se intentará utilizar pocas palabras técnicas, más sin embargo estas irán apareciendo para que después el usuario las pueda entender. Se incluirán varios ejemplos y ejercicios. | | |

Cuadro 4.11: Historia de usuario 7

4.5. Grafica de quemado

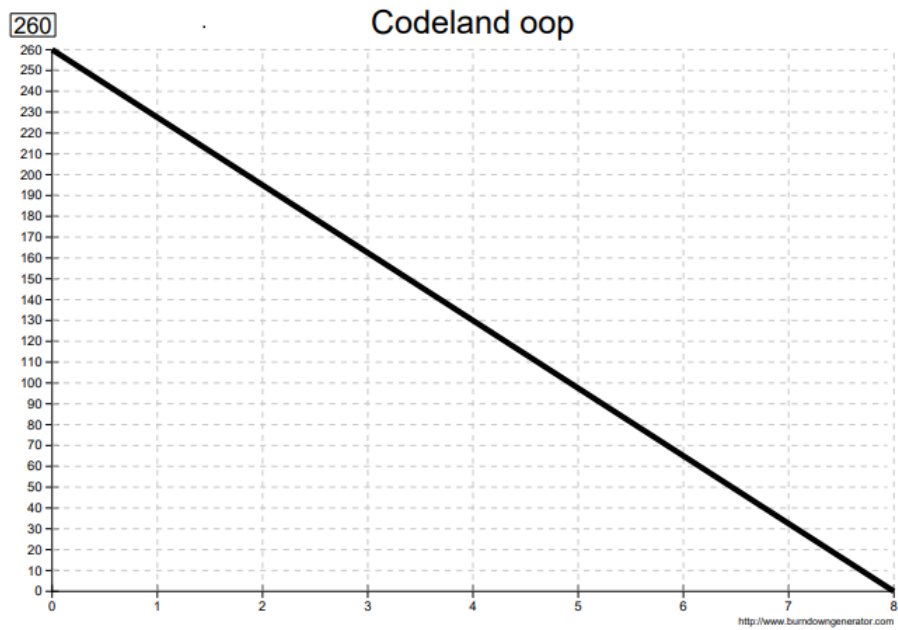


Figura 4.6: Grafica de quemado de las historias de usuario.

4.6. Maquetado

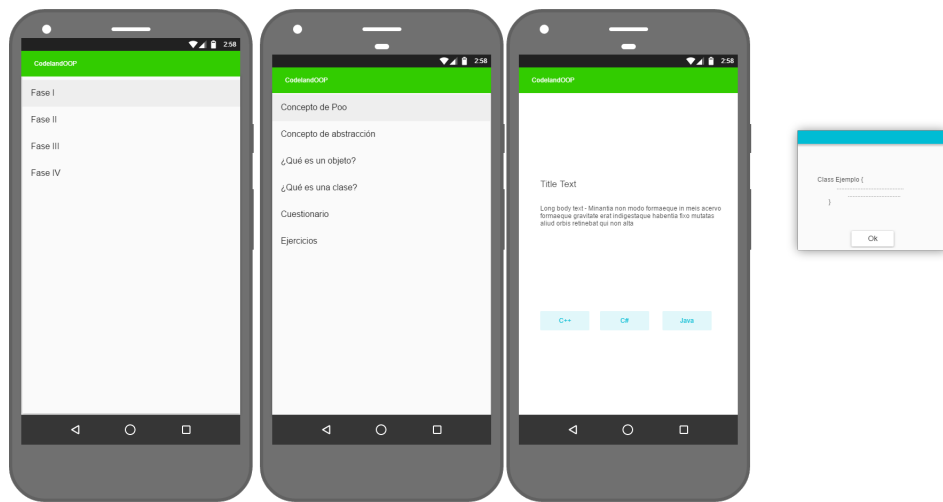


Figura 4.7: Pantallas de las fases, temas, tema, ejercicios y ejemplo.

Capítulo 5

Resultados

Se mostrarán las pantallas que se obtuvieron al desarrollar la aplicación.

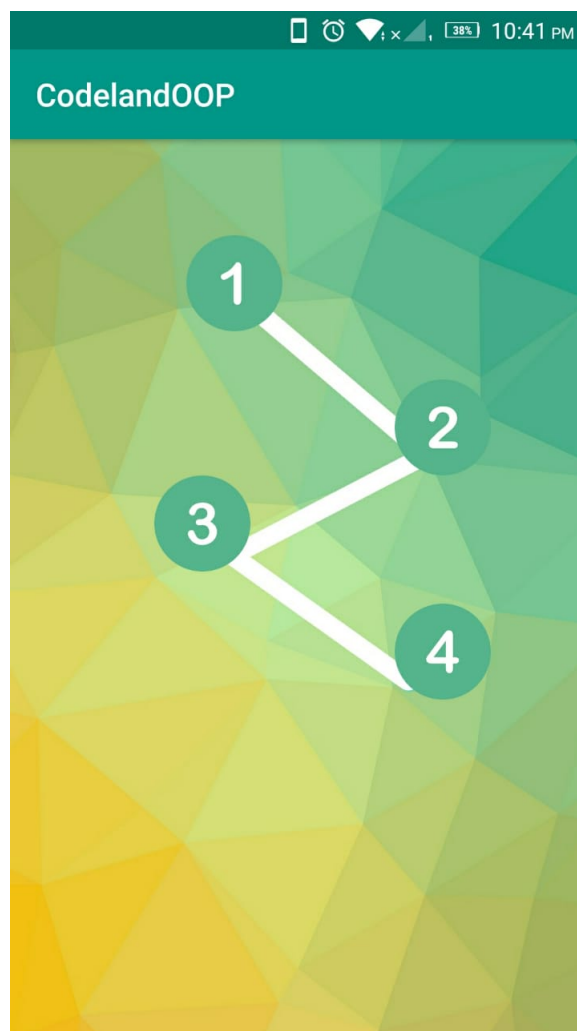


Figura 5.1: Pantalla de inicio donde se muestran las fases.



Figura 5.2: Pantalla de selección de los temas de cada fase.

Referencias

- Alvarez, M. and Alcázar, I. (2014). Introducción a git y github.
- Camacho, M. and Lara, T. (2011). M-learning en españa, portugal y américa latina. *Recuperado de: <http://scopeo.usal.es/sites/all/files/scopeom003.pdf>*.
- Dykes, G. and Knight, H. R. (2012). Mobile learning for teachers in europe: Exploring the potential of mobile technologies to support teachers and improve practice. *United Nations Educational, Scientific and Cultural Organization (UNESCO), UNESCO Working Paper Series on Mobile Learning, France*.
- Izquierdo, L. R. (2007). Introducción a la programación orientada a objetos. *Publicación online*.
- Joyanes Aguilar, L. (1996). *Programación orientada a objetos*. Number 004.652. 5. McGraw-Hill Interamericana,.
- Olivares Carmona, K. M., Angulo Armenta, J., Torres Gastelú, C. A., and Madrid García, E. M. (2016). Las tic en educación: metaanálisis sobre investigación y líneas emergentes en méxico. *Apertura (Guadalajara, Jal.)*, 8(2):100–115.
- Owens, M. and Allen, G. (2010). *SQLite*. Springer.
- Palacios, G. E. D., Gutiérrez, D., and Vargas, E. V. (2011). Tics en la educación.
- Spigariol, L. and Passerini, N. (2013). Enseñando a programar en la orientación a objetos. *UTN FRC, Córdoba, Argentina*.
- Studio, A. (2016). Conoce android studio. *Recuperado de: <https://developer.android.com/studio/index.html>*.
- Traxler, J. (2007). Defining, discussing and evaluating mobile learning: The moving finger writes and having writ.... *The International Review of Research in Open and Distributed Learning*, 8(2).
- Velarde de Barraza, O., Murillo de Velásquez, M., Gómez de Meléndez, L., and Castillo de Krol, F. (2006). Introducción a la programación orientada a objetos.

Anexos 6

Anexos

6.1. Plan de pruebas

El plan de pruebas es el documento fundamental en torno al cual se desarrollan las pruebas. Un plan de pruebas explica cómo el conjunto de pruebas debe ser usado y el soporte necesario para realizar el proceso de pruebas. Describe alcance, enfoque, recursos y calendarización de actividades de prueba. Identifica las tareas de prueba a desarrollar, los responsables de cada tarea y los riesgos asociados. Generalmente un plan de pruebas es un documento bastante extenso y debe organizarse para que sea entendible y mantenible. Para esto se propone el estándar IEEE 829 de Documentación el cual provee una tabla de contenidos para documentar un conjunto de pruebas.

| IEE 829 | Contenido |
|--------------------------------|--|
| Plan de Pruebas CodeLandOOP | <ul style="list-style-type: none">-Items a probarMenú FasesMenú TemasCuestionarioEjercicioEjemploInformación mostrada al usuario-Textbox-Labels-Combobox-BotonesCaracterísticas a ser probadas.-Que los botones realizaran las acciones para los que estaban diseñados-Que los Labels tuvieran una buena ortografía-Requerimientos funcionales-Requerimientos no funcionales-Que tenga configurada una buena tabulación.-Adaptable a cualquier resoluciónCaracterísticas que no se van a probar-Compatibilidad en IOS |

| | |
|----------------------------|---|
| Enfoque | <p>Este plan de pruebas es necesario para el aseguramiento de la calidad del sistema. Con este plan se seleccionan y se coordinan las actividades para asegurar la calidad del software durante el ciclo de vida del proyecto y aún después al ser entregado al cliente. Los objetivos que se pretenden alcanzar con la aplicación del plan de pruebas son</p> <p>las siguientes:</p> <ul style="list-style-type: none"> -Encontrar la mayor cantidad de errores como sea posible, ya sea tanto en los TextBox, como la ortografía que hay en las Labels, los botones, los ComboBox. -Supervisar si se cumple las especificaciones de diseño establecidas por el cliente. -Supervisar si se cumple los requisitos del análisis que se hicieron en la planificación del diseño y desarrollo del software. -Realizar pruebas las pruebas necesarias de rendimiento y capacidad del sistema. -Encontrar los problemas importantes y determinar los riesgos percibidos en cuanto a la calidad del producto. <p>Los tipos de pruebas que se realizarán al software son:</p> <ul style="list-style-type: none"> -Pruebas de Función -Pruebas de Interfaces de usuario -Pruebas de Desempeño |
| Casos de prueba | |
| Tareas y responsabilidades | <p>Administrador del plan de pruebas (análisis)</p> <ul style="list-style-type: none"> -Definición de casos de prueba -Resumen del ciclo de prueba (resultados y observaciones) <p>Desarrolladores o personal de ejecución de pruebas</p> <ul style="list-style-type: none"> -Pruebas de Función -Pruebas de Interfaces de usuario -Pruebas de Desempeño |
| Requerimientos de ambiente | <p>La aplicación se puede ejecutar perfectamente en un dispositivo que contenga un procesador equivalente a 2.6 Ghz y 256 MB en RAM.</p> <p>Se requiere que para la prueba se cuente con el SO Android en cualquiera de sus distintas versiones, así como la instalación de la aplicación CodeLandOOP.</p> |

| | |
|------------------------------------|---|
| Personal necesario y entrenamiento | <p>A nivel personal, se requiere atención por el detalle, concentración, organización y capacidad para analizar los resultados de las pruebas en busca de soluciones.</p> <p>Este trabajo implica la colaboración con otros profesionales, por lo que saber trabajar en equipo será fundamental.</p> <p>En cuanto a los aspectos técnicos cabe destacar que los lenguajes de programación que este profesional debería aprender para corregir errores y ejecutar pruebas variarán dependiendo del área en que se especialice.</p> <p>Sin embargo, el manejo de lenguaje Java y XML son necesarios en conjunto con Android Studio.</p> |
| Planificación | De acuerdo a la planificación descrita en el diagrama de gantt el tiempo para realizar las pruebas es de 4 días, el cual se extenderá en caso de encontrar problemas, al aplicar las medidas necesarias se planificará el siguiente ciclo de pruebas a aplicar. |
| Riesgos y contingencias | Los problemas que se encuentren en el proceso serán documentados y al igual que los problemas se documentarán las soluciones de éstos, el proceso que se seguirá para alcanzar la resolución de dichos problemas será el ir identificando cada uno de los problemas y aplicar las medidas necesarias para la solución de éstos. |
| Aprobación | Encargado del plan de pruebas, realizará resumen del ciclo de prueba (resultados y observaciones), y dará su aprobación si se obtuvieron resultados positivos de todos los casos de prueba y no hubo errores sin resolver de Gravedad. Esto demuestra la consecución de los objetivos de las pruebas, con lo que la solución está lista para el lanzamiento. |

6.2. Casos de pruebas

| 1. Navegar entre las ventanas | Código del CP: 1 | Si / No |
|---|------------------|---------|
| Descripción: Validar que el botón te lleve a la ventana que debería ir. | | |
| Prerequisitos: Iniciar la aplicación. | | |
| Pasos: <ul style="list-style-type: none"> ▪ Dar click en un elemento de los menús. ▪ Ver que la pantalla que sale sea la esperada. ▪ Volver a repetir para los demás ítem | | |
| Resultado esperado: Al presionar un elemento dentro de la ventana este te deberá llevar a la pantalla que se espere. | | |
| Resultado obtenido: <Resultado esperado de la ejecución de la prueba> | | |

Cuadro 6.2: Caso de prueba número 1

| 2. Cargar ítems por fases | Código del CP: 2 | Si / No |
|---|------------------|---------|
| Descripción: Se validará que se despliegue el contenido que se espera dependiendo de la fase que sea seleccionada. | | |
| Prerequisitos: Listado de temas por fase. | | |
| Pasos: <ul style="list-style-type: none"> ▪ Seleccionar una fase. ▪ Verificar que los ítems que se muestren pertenezcan a esa fase. ▪ Regresar al menú. ▪ Repetir los pasos anteriores para cada fase. | | |
| Resultado esperado: Los temas mostrados pertenecen a la fase seleccionada. | | |
| Resultado obtenido: <Resultado esperado de la ejecución de la prueba> | | |

Cuadro 6.3: Caso de prueba número 2

| 3. Cargar contenidos por temas | Código del CP: 3 | Si / No |
|---|------------------|---------|
| Descripción: Validar que se despliegue el contenido que se espera dependiendo el tema seleccionado. | | |
| Prerequisitos: Seleccionar una fase. | | |
| Pasos: <ul style="list-style-type: none"> ■ Seleccionar un tema. ■ Verificar que el contenido mostrado corresponda al tema. ■ Regresar al listado de temas. ■ Volver a repetir para cada tema. | | |
| Resultado esperado: Los contenidos mostrados pertenecen al tema seleccionado. | | |
| Resultado obtenido: <Resultado esperado de la ejecución de la prueba> | | |

Cuadro 6.4: Caso de prueba numero 3

| 4. Mostrar información completa | Código del CP: 3 | Si / No |
|---|------------------|---------|
| Descripción: Se validará que se la información que se muestra sea toda la que debería aparecer. | | |
| Prerequisitos: Información de los temas. | | |
| Pasos: <ul style="list-style-type: none"> ■ Seleccionar un tema. ■ Verificar que el contenido mostrado no carezca de información. ■ Regresar al listado de temas. ■ Volver a repetir para cada tema. | | |
| Resultado esperado: La información mostrada es la misma que debería mostrarse. | | |
| Resultado obtenido: <Resultado esperado de la ejecución de la prueba> | | |

Cuadro 6.5: Caso de prueba numero 4

| 5. Mostrar correctamente los tamaños | Código del CP: 5 | Si / No |
|---|------------------|---------|
| Descripción: Se validará que se la información que se muestra sea toda la que debería aparecer. | | |
| Prerequisitos: Dispositivos con diferentes tamaños de pantalla. | | |
| Pasos: <ul style="list-style-type: none"> ■ Instalar la apk. ■ Navegar por las distintas pantallas. ■ Verificar que en todas las pantallas se muestren correctamente iconos, botones, ítems, entre otros. | | |
| Resultado esperado: Los iconos y demás elementos se encuentran en los lugares esperados. | | |
| Resultado obtenido: <Resultado esperado de la ejecución de la prueba> | | |

Cuadro 6.6: Caso de prueba numero 5