TEC | Tecnológico de Costa Rica

# Informe Final de Proyecto

# Arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles.

## Documento 1


Investigadores:

Jeff Schmidt Peralta
Abel Méndez Porras
Andrei Fuentes Leiva
María Estrada Sánchez
Adriana Álvarez Figueroa


Mayo de 2015

## Tabla de contenido

## 3. Título.

Arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles.

## 4. Autores y direcciones.

Ing. Jeff Schmidt Peralta, MBA
-Coordinador del proyecto
jschmidtcr@gmail.com

Máster Abel Méndez Porras
-Investigador
mendez.abel@gmail.com

Máster Andrei Fuentes Leiva
-Investigador
mendez.abel@gmail.com

Máster María Estrada Sánchez
-Investigadora
mariaestrada.s@gmail.com

Máster Adriana Álvarez Figueroa
-Investigadora
adriana.alvarezf@gmail.com

## 5. Resumen

El cambio constante en la tecnología afecta las organizaciones y la sociedad en general. Esto provoca el surgimiento de nuevos paradigmas ante el auge constante de la ciencia y la tecnología en campos como la educación.
La explosión de nuevas plataformas computacionales, especialmente en dispositivos móviles, motivan la generación de metodologías que permitan el desarrollo de aplicaciones para esas plataformas.
Este proyecto fue planteado para diseñar e implementar una arquitectura que permita desarrollar aplicaciones educativas para dispositivos móviles, que utilizan los sistemas operativos iOS y Android, desde una perspectiva de ingeniería.
La definición de una metodología y la consideración de diferentes aspectos para definir la arquitectura de aplicaciones móviles, es validada con el desarrollo de dos aplicaciones de forma exitosa.

## 6. Palabras clave.

Arquitectura, metodología de desarrollo, dispositivo móvil, plataforma.

## 7. Introducción

En la sociedad del conocimiento y con la globalización como elemento impostergable de nuestra sociedad, es indispensable que los paradigmas educativos se adapten a los cambios. El desarrollo de aplicaciones educativas o apps es un hecho, ya existen miles de opciones.

Los dispositivos móviles fueron introducidos desde los años 70 del siglo pasado, especialmente para facilitar las comunicaciones. Con el transcurso del tiempo, se introdujeron diversos dispositivos de computación móvil. En los años siguientes los teléfonos móviles fueron evolucionando, hasta fusionarse con los PDA (Personal Digital Assistant) y en la actualidad los smartphones o teléfonos inteligentes, son computadores móviles con capacidades de multimedia, transmisión de datos y acceso a redes como Internet, lo cual ha explosionado su mercado en forma exponencial.

El propósito principal del proyecto consiste en desarrollar una arquitectura para el desarrollo de aplicaciones educativas utilizando dispositivos móviles, en sus dos entornos tecnológicos más representativos: iOS y Android. Esta arquitectura va a permitir crear juegos y aplicaciones de primer nivel, sirvan como complemento a la educación formal y que a la vez promuevan el aprendizaje autodidacta.

El abordaje del desarrollo de aplicaciones educativas móviles desde la perspectiva de ingeniería de software, es de gran importancia. Con el auge de la computación móvil, es importante sistematizar su desarrollo y estructurar el diseño de las aplicaciones, en especial las orientadas a fortalecer la ejecución.

La definición de una arquitectura en el desarrollo de aplicaciones es una tarea compleja, los beneficios que pueden obtenerse de contar con esa arquitectura, permitiría desarrollar apps en tiempo menor y cumpliendo con estándares que aseguren su calidad.

El objetivo general de este proyecto es diseñar e implementar una arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles. Además el proyecto de investigación persigue lograr los siguientes objetivos específicos:

- Identificar y contrastar las principales tendencias internacionales en el aprovechamiento de las tecnologías digitales móviles para la educación.
- Identificar y contrastar las principales metodologías internacionales utilizadas para el desarrollo en dispositivos móviles.
- Definir los pasos de una arquitectura para el desarrollo de aplicaciones educativas en dispositivos móviles.
- Generar al menos dos aplicaciones educativas utilizando la metodología propuesta.

La metodología y la definición de una arquitectura general que se logró en este proyecto de investigación, es un primer avance en convertir el desarrollo de proyectos, que lleven a conseguir apps educativas de una manera más sistemática.

Para validar tanto la metodología como la arquitectura propuesta se desarrollaron dos proyectos, cada uno con una app. Se utilizó la metodología propuesta y se tomaron en cuenta las consideraciones de la arquitectura, para conseguir dos casos de éxito. Los detalles de las apps desarrolladas pueden obtenerse como anexo de este informe.

# 8. Marco Teórico

El desarrollo de aplicaciones en entornos móviles sufrió un cambio radical con la liberación del primer iPhone de la empresa Apple en 2007. A partir de este evento, tanto Apple, Google, Nokia, Blackberry y otras empresas crearon sus productos, dispararon sus ventas y por tanto, el negocio de desarrollar aplicaciones para estas plataformas, se volvió más atractivo, en la actualidad se cuenta con millones de programas (apps) para estos dispositivos. El canal de distribución por excelencia es Internet, a través de tiendas especializadas o MarketPlaces.

Las metodologías para el desarrollo de sistemas que se han utilizado en los últimos 20 años han estado adaptándose a la producción de software para dispositivos móviles. Algunas de las metodologías más utilizadas son:

- Cascada (Waterfall): consiste en dividir el proyecto en fases secuenciales, basadas en el ciclo de vida de los sistemas. El desarrollo se realiza en fases, dando énfasis al planeamiento, cronogramas, presupuestos e implementación del sistema.
- Ágil (Agile): está basada en el desarrollo iterative, en el cual todas las fases del ciclo del proyecto son desglosadas en partes más pequeñas. El aspecto principal son los requerimientos.
- Programación Extrema (eXtreme Programming): XP pretende conseguir la calidad del software ante los cambiantes requerimientos del usuario. Como un tipo de desarrollo ágil se intentan tener multiples ciclos cortos de desarrollo, con la finalidad de reducir el costo de cambios..
- Desarrollo de Rápida Acción (Rapid Action Development): RAD se basa en evitar planeación excesiva para proyectos urgentes, de forma que sea fácil adaptarse a los cambios en los requerimientos.

Las metodologías anteriores se basaron en el ciclo de vida de los sistemas y pensando en proyectos de software tradicionales. El ciclo de vida comprende las fases y que normalmente van desde las etapas de análisis y especificación de requerimientos, diseño, programación, pruebas e implementación.

Las empresas que desarrollan aplicaciones o apps para dispositivos móviles, han adoptado algunas de estas metodologías a los nuevos desafíos del desarrollo en la computación móvil, sin que existan a la fecha estándares en la industria. Algunas empresas como Apple y Google, dictan algunos lineamientos generales, pero distan de ser consideradas metodologías de desarrollo.

En cuanto al proceso de educación, la tecnología ha venido siendo parte importante desde hace varios años, por medio de la televisión, videos, libros electrónicos, páginas web, wikis, computadores, dispositivos móviles y hasta educación virtualizada, en todos los niveles de los ciclos educativos. UNESCO [13] define el papel de las TICs en la educación en que pueden contribuir al acceso universal, la igualdad en la instrucción, el ejercicio de la enseñanza y el aprendizaje de calidad y el desarrollo profesional de los docentes, así como a la gestión dirección y administración más eficientes del sistema educativo.

Según Chen [1], el m-Learning es un apoyo a los procesos educativos de carácter móvil, que necesiten de alta interactividad en el proceso de aprendizaje, con integración de contenidos y ubicuidad de actividades de aprendizaje Para los estudiantes, las herramientas de aprendizaje móvil resultan muy atractivas ya que se utilizan las mismas tecnologías utilizadas para entretenimiento como recursos de apoyo, con la gran ventaja de que el estudiante puede llevarlos consigo a donde vaya.

Parsons y otros [10] examinan m-learning e identifican cuatro aspectos a considerar en una aplicación educativa móvil, relacionados con las características del dispositivo, tales como la interfaz, el contexto de aprendizaje, las experiencias de ese aprendizaje, tales como la inclusión de juegos y los objetivos del aprendizaje, relacionados con las metas que se desean mejorar. Estos aspectos deben ser considerados en una metodología de m-learning.

Según UNESCO [14] una enorme cantidad de iniciativas están demostrando diferentes maneras en las cuáles los dispositivos móviles pueden ayudar a confrontar los cambios educacionales y buscar nuevas estrategias para el aprendizaje. Los diseñadores de proyectos de m-learning han compartido sus mejores prácticas y desarrollado nuevos modelos de implementación. Para el año 2013 se espera que UNESCO publique, en asociación con Nokia, Policy Guidelines for Mobile Learning.

El m-learning ha generado múltiples investigaciones, por ejemplo MOBILearn Project, financiado por 24 países y cuyo objetivo principal fue la definición de modelos de soporte teóricos y validaciones empíricas para la efectiva enseñanza aprendizaje, tutorías en ambientes móviles, diseño instruccional y desarrollo de contenidos para aprendizaje móvil.

El modelo educativo apoyado en el uso de dispositivos móviles se ha desarrollado en diferentes modelos de aprendizaje. En el caso de Clive Shepherd [12] se plantean tres usos del m-Learning:

- En la fase preparatoria, antes del aprendizaje utilizando los diagnósticos para conocer el estado inicial del estudiante.
- Como método de apoyo al estudiante como preparación para evaluaciones y repaso de contenidos.
- Como práctica del aprendizaje, con aplicación a problemas del mundo real.

En Costa Rica hay varias iniciativas impulsadas por la Fundación Omar Dengo, además de empresas como Intel con su programa de donación de computadoras classmate a escuelas, así como la empresa Perimercados y Fundación Jiménez & Tanzi, con un programa similar utilizando computadoras del programa OLPC (One laptop per Child).

Además algunas escuelas y colegios privados de nuestro país ya solicitan que los estudiantes lleven a los centros de estudio sus computadores portátiles o dispositivos de computación móviles.

El campo del desarrollo de software para la educación ha sido estudiado desde varias aristas e instituciones como el Ministerio de Educación, la Fundación Omar Dengo, la Facultad de Educación de la Universidad de Costa Rica y la Escuela de Matemática del Instituto Tecnológico de Costa Rica cuentan con proyectos para desarrollo de software.

El desarrollo de apps para dispositivos móviles debe considerar aspectos como accesibilidad según el grupo de edad al cual va dirigida, claridad, eficiencia para mantener la atención, interfaz gráfica agradable, sistemas de recompensas y muchos otros aspectos más.

El uso de la tecnología en el proceso de educación es todavía muy reciente, todavía no se ha podido medir su impacto económico, social y propiamente en el campo educativo. El fenómeno va a continuar creciendo y una manera de apoyarlo es definir una arquitectura para el desarrollo de aplicaciones que vayan a ser utilizadas en el proceso educativo.

La tecnología debe ser considerada como otro insumo de la educación, la cual debe centrarse en el estudiante y los resultados del aprendizaje. La cantidad y calidad de aplicaciones informáticas que apoyen los programas educativos oficiales es una parte importante del proceso de mejora de la educación.

La disminución de la brecha digital es una meta de una sociedad como la costarricense, una importante manera de colaborar de una institución de educación superior, es ayudar a definir las bases para que el proceso de romper la brecha, en este caso, la construcción de aplicaciones educativas, se haga siguiendo un modelo de ingeniería adaptado a nuestra realidad.

# 9. Metodología

El proyecto se desarrolló utilizando la metodología de la ingeniería de software:

**1. Análisis y evaluación de propuestas publicadas o desarrolladas anteriormente .**
Debido a la explosión de las tecnologías móviles, han surgido esfuerzos de utilizar propuestas de ingeniería de software de proyectos tradicionales y adoptarlas al desarrollo de aplicaciones móviles.

**2. Definición de una metodología de desarrollo**
Para cualquier desarrollo es importante tener una metodología, que permita aplicar en forma sistemática los pasos para desarrollar una aplicación móvil.

**3. Establecimiento de los requerimientos**
Se analizaron las características de las aplicaciones móviles a desarrollar.

**4. Diseño de una arquitectura.**
En la medida de lo posible, se definieron una serie de consideraciones de diseño orientadas a crear una arquitectura para desarrollar aplicaciones móviles educativas.

**5. Implementación de la metodología y la arquitectura para el desarrollo de dos aplicaciones.**
La metodología definida y las consideraciones de diseño definidas en el proyecto fueron implementadas en el desarrollo de dos aplicaciones móviles.

# 10. Resultados

## 10.1 Las metodologías de desarrollo.

Una metodología de desarrollo es el conjunto de procedimientos, técnicas, documentos y ayudas que se van a realizar para la construcción de software, en nuestro caso de apps para dispositivos móviles.

Una app es una aplicación nativa específica para un dispositivo móvil para ser ejecutada en una plataforma móvil. Normalmente son descargadas a través de Internet desde sitios y cargadas directamente en el dispositivo. Las apps pueden tomar ventaja de los características de los dispositivos móviles, tales como cámaras, geolocalización, micrófonos, parlantes, sensores de movimiento y otros.

Las metodologías más utilizadas en la actualidad, han sido desarrolladas basándose en el ciclo de vida de los sistemas y pensando en proyectos de software tradicionales. El ciclo de vida comprende las fases y que normalmente van desde las etapas de análisis y especificación de requerimientos, diseño, programación, pruebas e implementación.

Estas metodologías para el desarrollo de sistemas han estado adaptándose a la producción de software para dispositivos móviles. Algunas de las metodologías más utilizadas son:

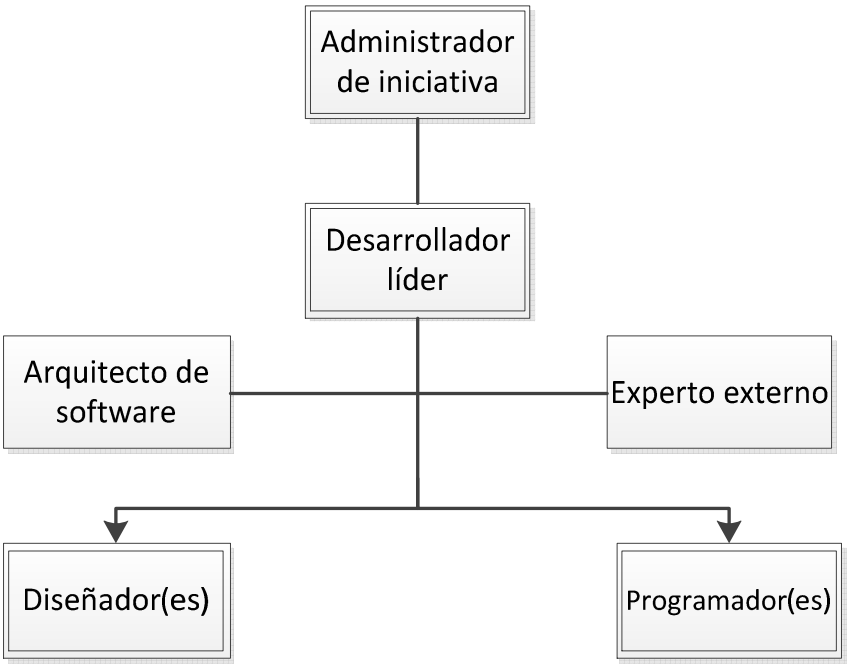
- Cascada (Waterfall): consiste en dividir el proyecto en fases secuenciales, basadas en el ciclo de vida de los sistemas. El desarrollo se realiza en fases, dando énfasis al planeamiento, cronogramas, presupuestos e implementación del sistema.
- Ágil (Agile): está basada en el desarrollo iterativo, en el cual todas las fases del ciclo del proyecto son desglosadas en partes más pequeñas. El aspecto principal son los requerimientos.
- Programación Extrema (Extreme Programming): XP pretende conseguir la calidad del software ante los cambiantes requerimientos del usuario. Como un tipo de desarrollo ágil se intentan tener múltiples ciclos cortos de desarrollo, con la finalidad de reducir el costo de cambios.
- Desarrollo de Rápida Acción (Rapid Action Development): RAD se basa en evitar planeación excesiva para proyectos urgentes, de forma que sea fácil adaptarse a los cambios en los requerimientos.
- Proceso Scrum: es un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto.

Las empresas que desarrollan apps han adoptado algunas de estas metodologías a los nuevos desafíos del desarrollo en la computación móvil, sin que existan a la fecha estándares en la industria.

En este proyecto de investigación se enfoca el desarrollo especialmente de aplicaciones educativas. El aprendizaje móvil (m-learning) permite que el acceso al conocimiento pueda darse en el momento adecuado, ya que la instrucción puede realizarse en cualquier momento y lugar, utilizando un dispositivo móvil. Bajo diversos paradigmas de la teoría del aprendizaje pueden construirse aplicaciones que apoyen y complementen los sistemas de educación.

## 10.2 Equipo de trabajo.

El equipo de trabajo encargado de un proyecto puede apreciarse en la siguiente figura:




Los roles dentro del proyecto se describen a continuación:

### 10.2.1 Administrador de iniciativa.

El administrador de la iniciativa es el reponsable del control del proyecto. En la fase de exploración e iniciación debe elaborar el Acta Constitutiva del Proyecto (ACP), documento que define de manera general la app a desarrollar, recursos requeridos y los principales hitos que van a regir el desarrollo.

### 10.2.2 Desarrollador líder.

El desarrollador líder es el encargado de liderar el equipo de desarrollo de una app. Participa en todas las fases del desarrollo y es el responsable de hacer el mejor uso de los recursos que se asignen.

### 10.2.3 Arquitecto de software.

El arquitecto de software tiene bajo su responsabilidad los repositorios de código y debe participar en la etapa de diseño del software, identificando componentes que puedan ser reutilizados del repositorio. Además debe identificar y documentar los nuevos componentes que van a irse incorporando al repositorio.

### 10.2.4 Experto externo.

El proyecto de investigación se centró en el desarrollo de apps educativas, en las cuales normalmente se requiere la participación de uno o más expertos externos, cuya labor principal es participar en el diseño de la app de forma que se incluyan los aspectos pedagógicos para que la app sea de calidad. Participa en todo el proceso de desarrollo, colaborando con el equipo de trabajo desde su perspectiva de experto en el tema que aborda la app.

### 10.2.5 Diseñadores.

Los diseñadores participan activamente en el proyecto desde sus fases iniciales y son los encargados de proponer y desarrollar el entorno gráfico de la app, así como las diferentes alternativas de interacción humano – dispositivo móvil.

### 10.2.6 Programadores.

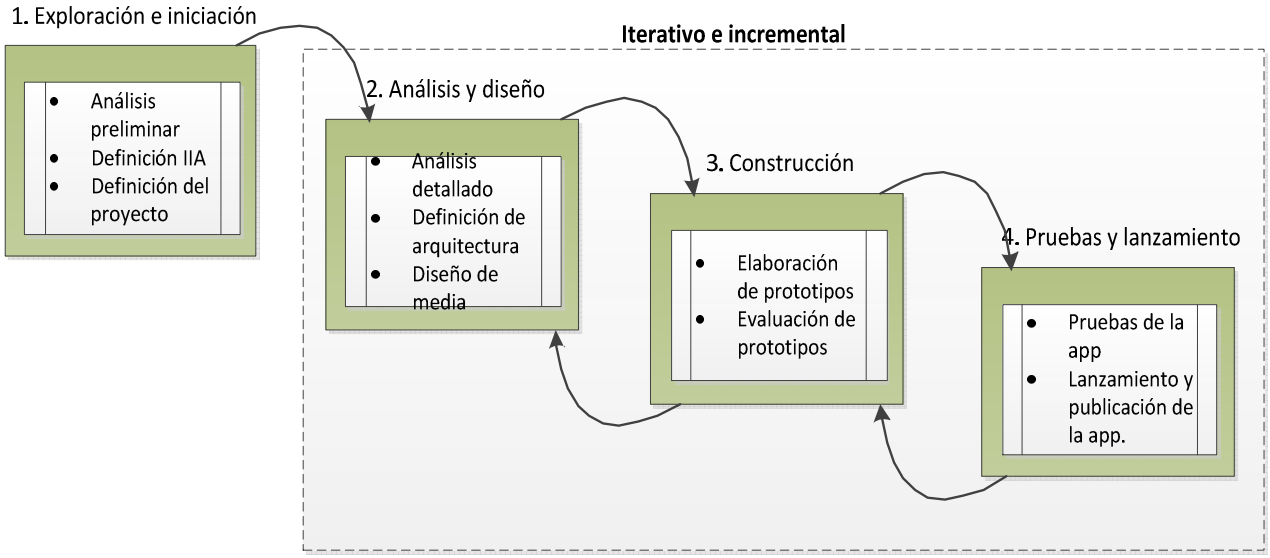
Los programadores junto con el desarrollador líder son los encargados de codificar los diferentes componentes de software que van a constituir la app. Además van a participar en las etapas de testing y lanzamiento.

## 10. 3 Descripción general de la metodología.

La metodología a utilizar que se propone en este proyecto de investigación, utiliza conceptos y buenas prácticas de las metodologías descritas anteriormente, basándose en los conceptos de un desarrollo iterativo e incremental.

Se hace énfasis en los resultados, para obtener un producto de calidad en el menor tiempo posible. La documentación inherente al proyecto no es exhaustiva, pero si la necesaria para poder darle continuidad.

Las fases del ciclo de vida de una app en esta metodología, pueden apreciarse en la siguiente figura:




Después de la primera fase en la cual se inicia el proceso de desarrollo, las tres fases siguientes deben ser iterativas. Las iteraciones deben ser pequeñas, de una a dos semanas, que permitan llevar el control del avance en el desarrollo.

Las etapas de las fases 2, 3 y 4 no son secuenciales, pueden traslaparse en el tiempo y pueden ser desarrolladas en paralelo.

### 10.3.1 FASE 1. Exploración e iniciación.

Esta fase se inicia con una idea inicial para una app. Esta idea debe ser analizada y en caso de determinarse su factibilidad a nivel técnico y de recursos, se documenta esta etapa inicial.

La fase de exploración e iniciación consta de las siguientes etapas:

**Análisis preliminar.**

En esta etapa se analiza la idea inicial de la app. Los miembros del equipo de desarrollo deben discutir el alcance que se desea alcanzar y desarrollar en forma paralela el documento de Idea Inicial de App.

**Definición de la idea inicial de la app.**

En esta primera etapa se debe definir de manera muy general la app a desarrollar. Se debe utilizar el documento Idea Inicial de App o IIA. El tiempo normal para esta etapa debería ser entre 1 y 2 semanas.

**Definición del proyecto.**

Una vez realizado el documento IIA, el administrador de la iniciativa es el responsable de elaborar una Acta Constitutiva del Proyecto (ACP), que permita conocer la visión del proyecto, los alcances y requerimientos de personal y otros recursos que se van a utilizar para poder desarrollarlo. Además se deben definir los hitos principales del proyecto. La duración de la elaboración del ACP no debe ser mayor a 1 semana.

### 10.3.2 FASE 2. Análisis y diseño.

En esta etapa se define toda la información que va a regir el desarrollo de la app. El producto de esta fase se debe resumir en el Documento de Diseño de App o DDA. Las etapas son iterativas, los cambios en cualquier parte del proceso deben documentarse en el DDA, el cual no debe verse como un documento estático, sino como la base para documentar este proceso iterativo.

El modelo se basa en realizar iteraciones o sprints semanales o quincenales, que permitan conocer los avances del proyecto. En estos puntos de control el desarrollador líder se reunirá con el administrador de la iniciativa, para conocer los avances en el proceso.

Las etapas de esta fase son:

**Análisis detallado.**

En esta etapa se deben definir los objetivos y el alcance de la app, en forma incremental, se inicia con ideas generales que se pueden ir aplicando en la Fase 3 Construcción, de tal forma que se vayan generando prototipos.

**Definición de arquitectura.**

Conforme se van conociendo los alcances de la app, se debe ir diseñando la arquitectura que va a tener la app, considerando la plataforma para la cual se va a desarrollar. La arquitectura puede cambiar según el avance en el proceso de desarrollo.

**Diseño de media.**

Los aspectos relacionados con la presentación de la app y la interfaz del usuario con la misma deben diseñarse en esta etapa. Se puede iniciar con bocetos o mockups, los cuales una vez aprobados por el equipo de trabajo pueden irse desarrollando formalmente.

### 10.3.3 FASE 3. Construcción.

En esta fase se construyen los componentes de software que van a conformar la app, considerando la utilización de los repositorios de código de la iniciativa. La codificación va dirigida a la elaboración de prototipos incrementales, a los cuales en cada sprint se le van agregando funcionalidades.

Las etapas de la fase de construcción son:

**Elaboración de prototipos.**

Conforme se avanza el análisis y diseño de la app, se deben ir construyendo los prototipos que van a ser evaluados en cada Sprint.

**Evaluación de prototipos.**

El equipo de desarrollo debe evaluar cada prototipo desarrollado, definiendo que aspectos deben mejorarse y cuáles van a ser los aspectos a incluir para el próximo sprint.

### 10.3.4 FASE 4. Pruebas y lanzamiento.

En esta fase se realiza el deployment de la app, se inician las pruebas hasta lograr un estado aceptable que permita obtener una primer reléase que pueda ser publicado.

Las etapas de esta fase de pruebas y lanzamiento son:

**Pruebas de la app.**

El testing de la app debe realizarse conforme se van liberando los prototipos, por parte de los miembros del equipo. Es importante probar diferentes tipos de dispositivos para asegurar la compatibilidad de la app.

Una vez que se ha conseguido una versión estable y que cumpla con los requerimientos definidos, es importante hacer una evaluación con una muestra de la población meta a la cual va dirigida la app, con la finalidad de obtener retroalimentación.

**Lanzamiento y publicación de la app.**

En la etapa de lanzamiento se debe definir la estrategia que va a regir la publicación de la app, considerando aspectos como divulgación, si se va a cobrar o no y en que sitios se va a promover.

## 10.4 Arquitectura para el desarrollo de aplicaciones móviles.

Esta sección expone la Arquitectura de Software en términos generales de una aplicación móvil educativa.

> *The software architecture of a program or computing system is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships between them.*

### 10.4.1 Consideraciones generales de diseño para aplicaciones móviles

Las siguientes guías pretenden proveer información sobre aspectos que deben ser estudiados al diseñar una aplicación móvil. Debido a información recolectada por un grupo Quality Assurance, resulta relevante para el contexto de GoTouch tomar en cuenta las consideraciones para lograr obtener un rendimiento y una eficiencia esperada de la aplicación.

- **Decidir si desarrollar un *rich client, thin web client* o una *rich internet application*.** Para estos efectos se debe saber que una aplicación móvil, por lo general, consta de varias capas que son: *user experience,* negocio y datos. *Rich client* significa que la capa de negocio y datos estará en el mismo dispositivo; mientras que *Thin client,* estas capas estarán en el servidor. En GoTouch, la mayoría de las aplicaciones actualmente no necesitan de estar conectadas a Internet ni en comunicación con ningún servidor, por lo que se trata mayormente de aplicaciones *Rich Client*. Por último, si la aplicación requiere una interfaz de usuario compleja, acceso limitado a recursos locales y debe ser portable a otras plataformas, lo recomendable sería diseñar un *Rich Internet Application* client.

- **Determinar el tipo de dispositivos que se soportará**. Considerar tamaño de pantalla, resolución (DPI), las características de rendimiento del CPU, memoria y capacidad de almacenamiento.

- **Tome en cuenta escenarios de poca conectividad y ancho de banda limitado**. Cuando la conectividad a redes es requerida, las aplicaciones deben manejar casos en los que la conexión a la red sea intermitente o no exista.

- **Diseñar una arquitectura en capas que sea apropiada para dispositivos móviles que promueva la reutilización y la mantenibilidad**. Utilizar el concepto de capas ayuda a maximizar la separación de responsabilidades i.e. cada capa tiene una función específica y separada del resto. El diseño debe ser simplificado en comparación con una aplicación de escritorio o web, debido a las capacidades del dispositivo móvil

- **Considere siempre limitaciones de recursos del dispositivo como duración de la batería, tamaño de la memoria y velocidad del procesador**. Todas las decisiones de diseño deben tomar en cuenta estos recursos. Usualmente la batería es uno de los factores más limitantes. *Backlighting* (luz de la pantalla), leer y escribir en memoria, conexiones inalámbricas, hardware especializado y velocidad del procesador tienen un impacto en el uso general del poder.

### 10.4.2 Áreas específicas de consideración

Existen varios asuntos comunes que se deben considerar cuando se está diseñando la aplicación. Se categorizarán en áreas específicas del diseño que son:

- Comunicación: La comunicación incluye comunicación inalámbrica (*over the air*) y comunicación alámbrica con un host PC, al igual que comunicación más especializada como Bluetooth o Infrarrojo (Infrared Data Association).

- Gestión de la configuración: Cuando se diseñe la gestión de la configuración de los dispositivos, se debe considerar cómo manejar *resets* del dispositivo.
- Acceso a datos: En el momento de diseñar el acceso a datos, considere cómo el bajo ancho de banda, la alta latencia y la intermitencia en la conectividad impactarán su diseño.
- Manejo de excepciones: Un buen manejo de excepciones previene que se muestren mensajes con información sensible al usuario, brinda robustez a la aplicación y ayuda a evitar un estado inconsistente en caso de un error.
- Gestión del poder: todas las decisiones de diseño deben tomar en consideración cuánto poder consume el dispositivo y su efecto en la duración de la batería.
- Pruebas: El *debuggear* (depurar) las aplicaciones móviles puede ser mucho más costoso que hacerlo con una aplicación similar en una PC. Se debe considerar este costo cuando se decida cuáles y cuántos dispositivos soportará la aplicación.
- Dispositivo: Se podría estar apuntando a varios tipos de dispositivo para una sola aplicación. Se debe tener presente la heterogeneidad en los distintos dispositivos a la hora de diseñar la aplicación. Factores a tomar en cuenta como tamaño de la pantalla y orientación, limitaciones en memoria y espacio de almacenamiento, ancho de banda y conectividad.
- Validación: Validar los datos de entrada antes de enviarlos a un servidor puede reducir el *overhead* y mejorar el rendimiento. Esto también hace que la aplicación sea más responsiva cuando el usuario ingrese datos inválidos.

### 10.4.3 Capa de Presentación

Contiene los componentes encargados de implementar y desplegar la interfaz de usuario. Sumado a esto también se encarga de manejar la interacción que tenga el usuario con la interfaz. Esta capa a su vez, es la encargada de controlar la entrada de datos por parte de los usuarios.

Los componentes de la Capa de Presentación son:

- **Componentes de interfaz con el usuario (UI)**
  Son los componentes que le permiten al usuario interactuar con la aplicación. Además de ser los encargados de *renderizar* y darle formato a los datos que son mostrados a los usuarios de la aplicación. A su vez se deben encargar de validar los datos que son introducidos por los usuarios.

- **Componentes de proceso**
  Son los encargados de sincronizar, orquestar y organizar las interacciones del usuario. En muchos casos separar estos componentes resulta beneficioso, cuando se tiene una UI muy compleja.

#### 10.4.3.1 Enfoque

Descripción de los pasos a seguir cuando se va a diseñar la capa de presentación. Estos pasos garantizan que se vayan a considerar todos los aspectos relevantes para el desarrollo de la arquitectura, como tipo de cliente, modo de mostrar los datos, estrategia de validación de datos, estrategia lógica de negocio y estrategia para comunicarse con otras capas.

#### 10.4.3.2 Consideraciones de Diseño

Factores críticos de éxito que se deben considerar cuando se diseña la capa de presentación.

- *Elegir la apropiada tecnología para elaborar la IU*
- *Utilizar patrones*
- *Diseñar de manera separada los componentes de UI con los componentes de proceso*

#### 10.4.3.3 Almacenamiento en Caché

El almacenamiento en cache es una de los mejores mecanismos para mejorar el desempeño de las aplicaciones. Pero estos datos almacenados en cache deben tener un objetivo bien definido y no almacenar todo por almacenarlo. Aspectos a considerar a la hora de diseñar la estrategia de almacenamiento en cache:

- Los recursos siempre son limitados
- No almacenar en cache datos volátiles.
- Almacenar datos que estén listos para usarse, y no datos que deben ser procesados
- No almacenar en cache datos sensibles a menos que estén encriptados.
- Que la aplicación no dependa de la información almacenada en la cache

#### 10.4.3.4 Administración de Excepciones

Utilizar un mecanismo centralizado que se encargue que capturar y responder a las excepciones de manera consistente. Se debe prestar especial atención a las excepciones que se propagan a través de las capas o de sus límites. Diseñar las excepciones para que no impacten a la aplicación ni expongan información sensible.

- Utilizar mensajes amigables para notificar al usuario que ha ocurrido un error.
- No se debe exponer datos importantes en las páginas de error
- Diseñar una excepción global que se encargue de mostrar los mensajes de error
- No utilizar excepciones para controlar la lógica de la aplicación.

#### 10.4.3.5 Input

Diseñar el método de ingreso de datos, en los requerimientos de la aplicación. Para tener un mayor grado de usabilidad, se recomienda utilizar mejores prácticas establecidas por la industria.

- Utilizar formularios que controlen los accesos de datos, en las tareas de recolección de datos.
- Utilice un asistente, para el usuario, en las tareas de recolección de datos más complejas.
- Considerar la accesibilidad en el diseño, considerar personas con discapacidades.

#### 10.4.3.6 Layout

Diseñar la interfaz de tal forma que los componentes sean independientes de los procesos de la UI. Para esto se busca diseñar una interfaz que no necesite código de procesos necesariamente, sino aspectos centrados únicamente en el despliegue de la interfaz. Considerar también el dispositivo, dado que las pantallas de los móviles son pequeñas, no se puede mostrar en una sola todas las funcionalidades, es por esto que se torna importante la utilización de patrones. Se debe buscar que la interfaz sea lo más consistente posible, así como la aplicación.

- Utilizar plantillas que le brinden a la aplicación una apariencia similar en todas sus pantallas
- Utilizar un diseño en común en toda la aplicación para facilitar la accesibilidad y usabilidad.
- Considerar los métodos que tiene el móvil para la interacción como touch screen.
- Utilizar patrones para separar la interfaz de usuario con los procesos de esta.

#### 10.4.3.7 Navegar

Desarrollar la estrategia de navegación de tal forma que le permita al usuario navegar de manera fácil a través de las diferentes pantallas o páginas. Asegurarse que los controles y enlaces través de la aplicación son consistentes, para evitar la confusión del usuario.

- Utilizar patrones de diseño conocidos para separar la UI de la lógica de navegación
- Una barra de tareas y un menú bien diseñado ayudan a encontrar funcionalidades de la UI.
- Hacer de la navegación lo más predecible que se pueda.
- Determinar como la aplicación va a preservar el estado de la navegación

#### 10.4.3.8 Procesamiento de Solicitudes

Diseñar el procesamiento de solicitudes, el código de mantenimiento y las pruebas con la respuesta del cliente en mente.

- Utilizar operaciones asincrónicas o subprocesos de trabajo para evitar que se bloquee la interfaz cuando se realicen tareas de larga duración.

#### 10.4.3.9 Experiencia del Usuario.

Tomar en cuenta estudios de usabilidad, entrevistas y encuestas que le permitan entender que quiere el usuario y que espera de la aplicación. Desarrolle la UI con los resultados de estos estudios en mente

- No desarrollar interfaces demasiado sobrecargadas o complejas.
- Se debe diseñar de tal manera que se soporte la personalización por parte del usuario.
- Permitir que el usuario controle como desea interactuar con la aplicación

#### 10.4.3.10 Componentes de UI

Son los controles y componentes utilizados para mostrar la información al usuario y aceptar los datos ingresados por este.

- Crear controles personalizados.
- Buscar ampliar las funciones de los controles existentes.

#### 10.4.3.11 Componentes de procesamiento de UI

Sincronizan las interacciones del usuario. No son siempre necesarias, se utilizan solo en el caso en que se necesita un rendimiento significante a la hora de procesar la capa de presentación. Se debe tener el cuidado de no mezclar la lógica del negocio con la lógica de presentación

- No crear componentes de procesamiento a menos que sean necesarios
- En caso de que la UI necesite de procesamiento complejo o requiera comunicarse con otras capas, usar estos componentes para desacoplar este procesamiento de la interfaz de usuario.
- Dividir los componentes de procesos en tres roles distintos: modelo, vista, controlador
- Utilizar patrones.

#### 10.4.3.12 Validación

Diseñar una efectiva validación de los datos ingresados es una estrategia crítica para la seguridad de la aplicación. Genera las reglar necesarias para este proceso de validación, y que se vayan a utilizar en futuras aplicaciones.

- Validar todos los datos ingresados para reducir errores.
- Diseñar la estrategia de validación para restringir y limpiar las entradas maliciosas de datos.

#### 10.4.3.13 Guía de Patrones

Los patrones están divididos en las mismas categorías ya reseñadas. Se recomienda considerar estos patrones cuando se vaya a diseñar la capa de presentación:

- Llamada asincrónica (Asynchronous Callback)
- Dependencia de la Cache (Cache Dependency)
- Cadena de responsabilidades (Chain of Responsibility)
- Patrón de comando (Command Pattern)
- Entidad de traducción (Entity Translator)
- Escudo de excepción (Exception Shielding)
- MVC (Model-View-Controller)
- Control de página (Page Controller)
- Modelo de Presentación (Presentation Model)
- Plantillas de vista (Template View)

#### 10.4.3.14 Mapeo de los Patrones con las categorías

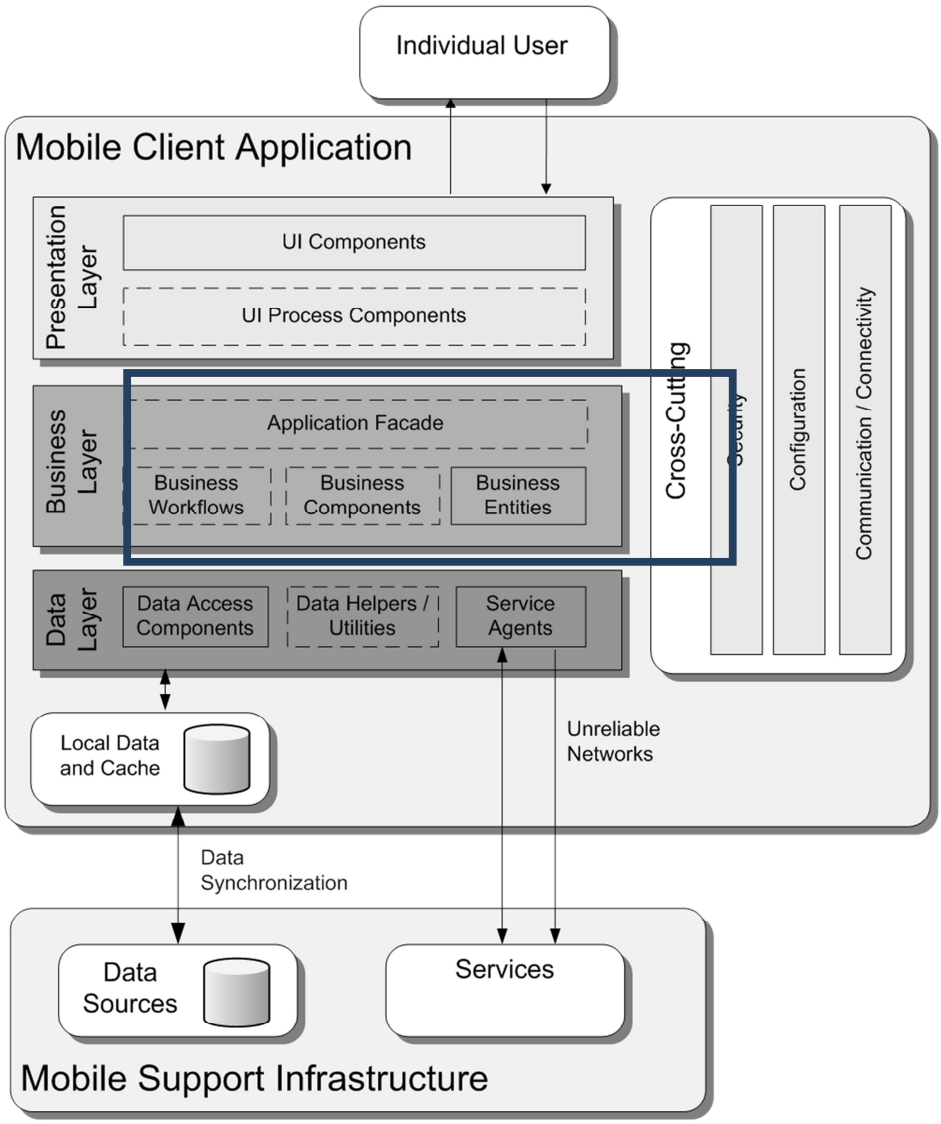
En la siguiente tabla se encuentra una relación que muestra cuáles patrones, de los presentados anteriormente, pueden ser usados en las categorías explicadas a lo largo del documento.

| Categoría | Patrón relacionado |
|---|---|
| **Almacenamiento en Caché** | Cache Dependency |
| **Administración de Excepciones** | Exception Shielding |
| **Layout** | Template View |
| **Navegar** | Command Pattern |
| **Presentación** | Entity Translator |
| **Experiencia del Usuario** | Asynchronous Callback<br>Chain of Responsibility |
| **Componentes de procesamiento de UI** | Model-View-Controller (MVC)<br>Presentation Model |

### 10.4.4 Capa de Negocio

Esta capa implementa la funcionalidad básica del sistema, y encapsula la lógica de negocio relevante. Por lo general, se compone de varios componentes, algunos de los cuales pueden exponer interfaces de servicio que otras pueden llamar. Para ciertas aplicaciones que requieren acceso a datos y servicios externos remotos, estos son consumidos por la capa de negocio a través de servicios Web.

La siguiente imagen representa como la capa de negocio interactúa con las demás capas, así como los componentes mínimos que debe tener.




#### 10.4.4.1 Componentes clave del negocio

A continuación se describen los roles y responsabilidades de los principales componentes dentro de la capa de negocio.

1. **Fachada de la aplicación (opcional):** combina múltiples operaciones de negocio en una sola operación basada en mensajes. Es posible acceder a la fachada de la aplicación a través de la capa de presentación mediante el uso de diferentes tecnologías de comunicación.
2. **Componentes de negocio:** Dentro de la capa de negocio hay diferentes componentes que proporcionan servicios de negocio, tales como el procesamiento de reglas de negocio e formas de interactuar con los componentes de acceso a datos.

3. **Las entidades de negocio:** Los componentes de negocio que se utilizan para pasar los datos entre los demás componentes de negocio son considerados entidades de negocio. Los datos se pueden representar entidades de negocio del mundo real, tales como productos y pedidos, o las entidades de base de datos, como tablas y vistas. Las entidades empresariales se pueden implementar utilizando estructuras de datos tales como bases de datos y lenguaje de marcado extensible (XML).
4. **Flujos de trabajo de negocio:** Muchos procesos de negocio implican varios pasos que se deben realizar en el orden correcto y orquestado. Flujos de trabajo empresariales definen y coordinan los procesos empresariales de larga ejecución, de múltiples pasos, y pueden ser implementados utilizando herramientas de gestión de procesos de negocio.

Con el fin de minimizar la huella en el dispositivo, las aplicaciones móviles generalmente usan enfoques de estratificación menos rígida y un menor número de componentes discretos.

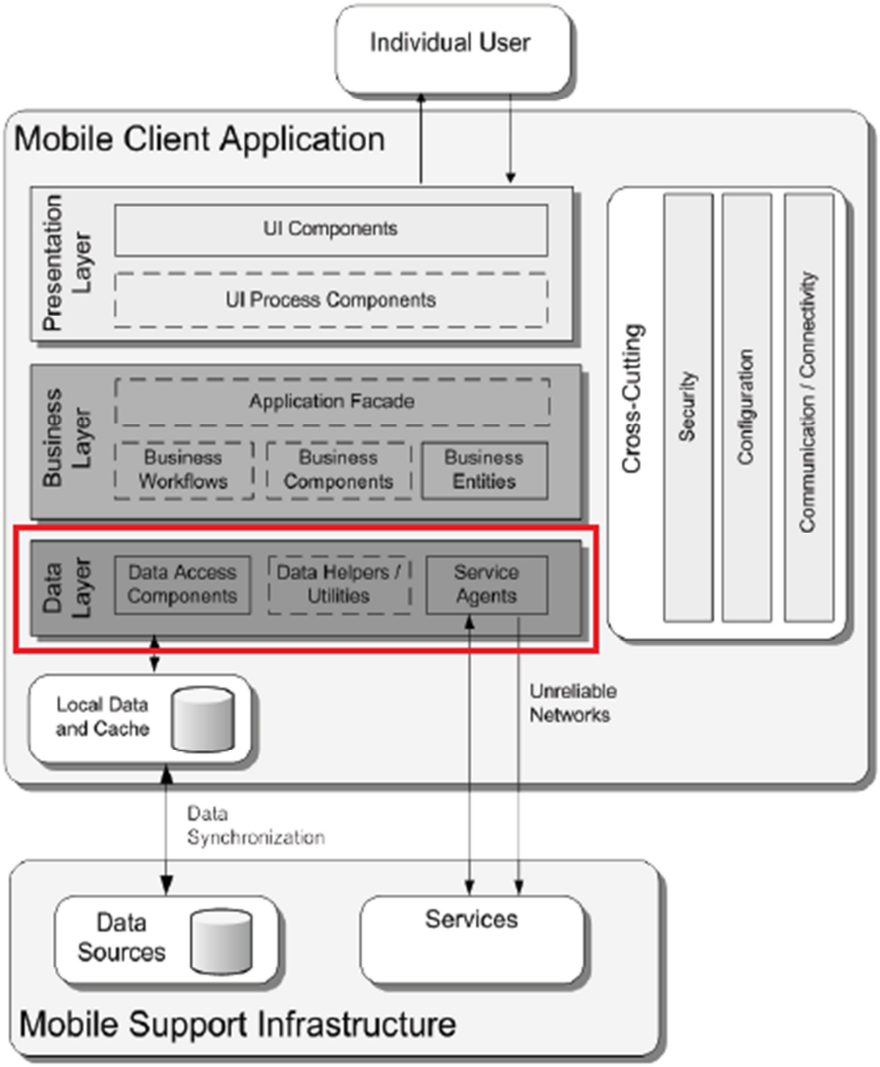
#### 10.4.4.2 Mapeo de Patrones a utilizar

Patrones principales están organizados por las principales categorías detalladas en la tabla siguiente. Considere el uso de estos patrones para tomar decisiones de diseño para cada categoría.

| Categoría | Patrones Relevantes |
|---|---|
| **Componentes de Negocio** | Fachada de aplicacion<br>Cadena de responsabilidad<br>Comando |
| **Entidades de negocio** | Entity Translator<br>Table Module |
| **Concurrencia y Transacciones** | Capture Transaction Details<br>Coarse-Grained Lock<br>Implicit Lock<br>Optimistic Offline Lock<br>Pessimistic Offline Lock<br>Transaction Script |
| **Acceso a Datos** | Active Record<br>Query Object<br>Row Data Gateway<br>Table Data Gateway |
| **Flujos de Trabajo** | Flujo de trabajo basada en datos<br>Flujo de trabajo humano<br>Flujo de trabajo secuencial<br>State-driven workflow |

### 10.4.5 Capa de Acceso a Datos

La siguiente figura muestra como la capa de datos se ajusta a la arquitectura típica de una aplicación.




#### 10.4.5.1 Componentes de la Capa de Datos

**Componentes lógicos del acceso a datos**: los componentes de acceso a datos abstraen la lógica necesaria para acceder a los almacenes de datos subyacentes. Centraliza la funcionalidad de acceso a datos, lo que hace que la aplicación sea más fácil de mantener y configurar.

**Componentes de ayuda o servicios:** la mayoría de las tareas de acceso a datos requieren cierta lógica común que puede ser extraída e implementada en un componente separado y reutilizable. Esto ayuda a simplificar la complejidad de los componentes de acceso a datos y sobre todo, minimiza el volumen de código a mantener.

Se componen de bibliotecas especializadas y rutinas personalizadas especialmente diseñadas para maximizar el rendimiento de acceso a datos y reducir el desarrollo de requerimientos de los componentes lógicos y las partes de agente de servicio de la capa.

**Agentes de servicio:** cuando un componente de negocio debe usar funcionalidad expuesta por un servicio externo (servicios brindados por un proveedor externo a la organización) como por ejemplo el servicio de correo de Google, se necesitara crear código que administre la semántica de comunicación con ese servicio. Los Agentes de Servicios aíslan dicha idiosincrasia de forma que, manteniendo ciertas interfaces, sería posible sustituir el servicio externo original por un segundo servicio diferente sin que nuestro sistema se vea afectado.

#### 10.4.5.2 Guía para diseñar la capa de acceso a datos

Un correcto enfoque para diseñar la capa de datos reducirá el tiempo de desarrollo y el mantenimiento de la capa de datos después de que la aplicación es desarrollada. Esta sección describe brevemente un enfoque de diseño efectivo para la capa de datos. Desarrollar las siguientes actividades claves cuando diseñamos la capa de datos:

1. **Crear un diseño global de la capa de acceso a datos**
2. **Diseñe los componentes de acceso a datos**
3. **Diseñe los componentes de ayuda o servicios**
4. **Diseñar los agentes de servicio**

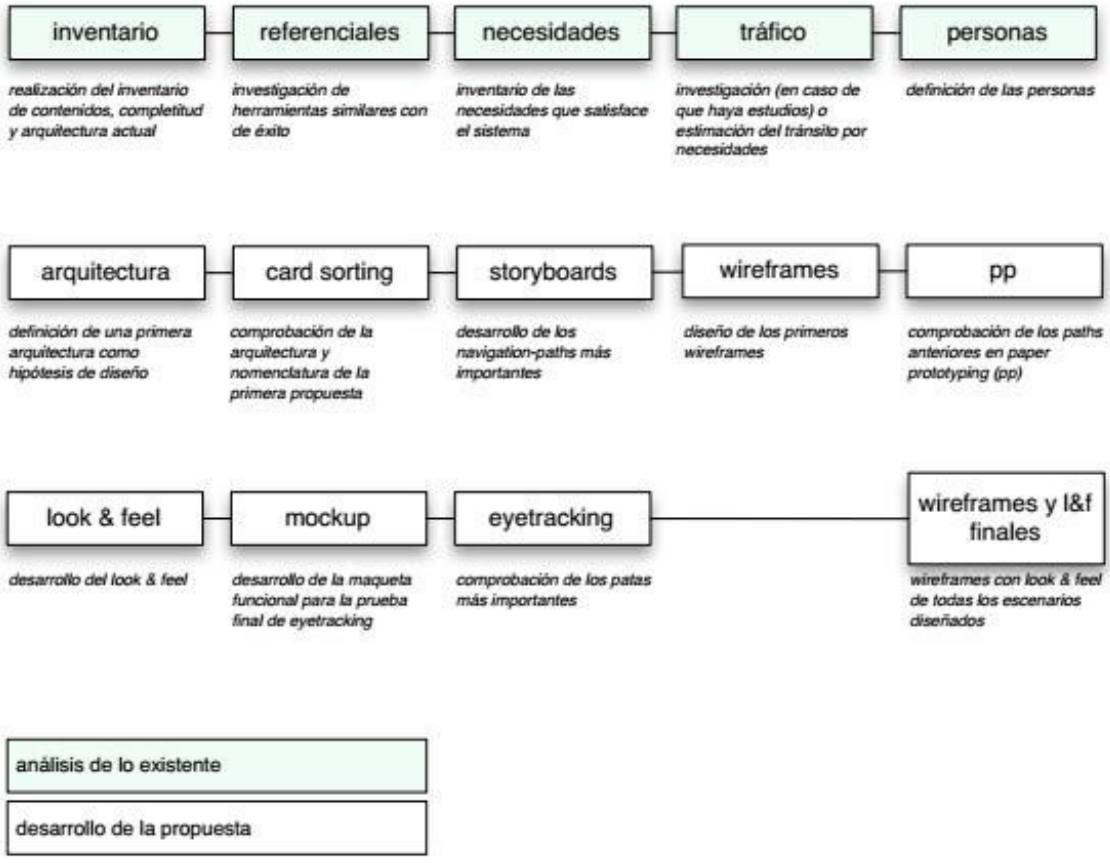
## 10.5 Arquitectura desde el punto de vista del Diseño de la App.

La arquitectura de una app desde el punto de vista del diseño, de la UI (User Interface), debe considerar los siguientes pasos:

- Definir cómo el usuario va usar la aplicación
- Consolidar una experiencia (UX) disfrutable para el usuario

- Desarrollo del storyboard: se específican las decisiones tomadas, en forma de storyboard, mostrando el flujo de las acciones a través de imágenes, para así especificar cómo se navega de un escenario a otro.
- Paper prototyping: se realiza una prueba simple de usabilidad antes de llegar a la etapa de desarrollo; el paperprototyping utiliza los wireframes que se definieron en el storyboard.
- Diseño interactivo: definir el aspecto gráfico de la herramienta de forma interactiva y con posibilidad de realizar cambios en forma gráfica.

El siguiente diagrama muestra los pasos a considerar para definir la arquitectura de una app desde el punto de vista de diseño.

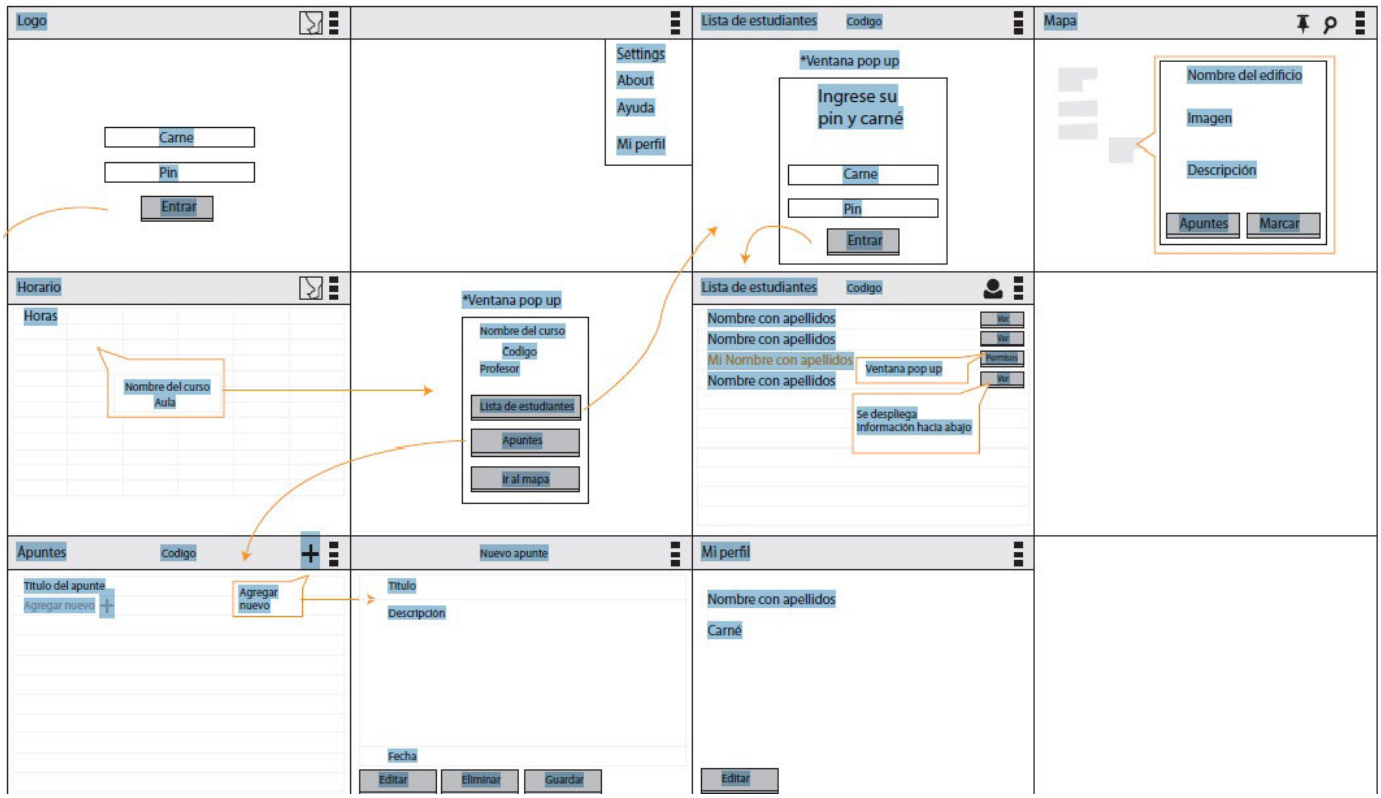


## 10.6 Arquitectura de información

La arquitectura de información es una forma de organizar el contenido y funciones de toda la aplicación, de forma que puedan ser encontrados rápidamente por el usuario. En sentido global, la arquitectura de información considera la relación entre los contenidos de diferentes pantallas y a nivel particular, la organización de contenidos dentro de la misma pantalla.

Luego de haber definido el Viaje de usuario y las funciones de la app, un diagrama de arquitectura utiliza esta información para determinar cuáles son las pantallas necesarias en cada etapa de ese viaje y qué funciones debería tener cada una de ellas.

Una forma de visualizar la arquitectura consiste en representar cada pantalla con un rectángulo donde las conexiones indican la forma de navegar de una pantalla a otra y a través de qué acción.




Un diagrama de arquitectura de información permite visualizar rápidamente los vínculos entre contenidos. Este diagrama sirve para estudiar la complejidad de la aplicación de un vistazo, analizar los diferentes niveles de profundidad, visualizar y entender la relación entre contenidos de una manera más organizada. Además, lo que se defina en esta etapa tendrá una incidencia directa en el tipo de navegación que se elija posteriormente.

# 11. Discusión y conclusiones

Hay varios enfoques de arquitectura que podrían satisfacer los requisitos, y elegir el diseño más adecuado significa la evaluación de varios factores, algunos de los cuales son únicos para el desarrollo móvil. Algunos de los factores comúnmente considerados son las plataformas de despliegue, los dispositivos específicos y perfiles de usuario, los contextos en los que es más probable que se utilice la aplicación, así como cualquier utilidad fuera de línea y perfiles de conectividad que la aplicación debe soportar.

La complejidad del flujo de trabajo y la riqueza de la experiencia del usuario que se requiere es probablemente uno de los factores más importantes que determinan esta elección. La elección de la arquitectura, sin duda, tendrá consecuencias a largo plazo, y los arquitectos de aplicaciones móviles que entender la visión del cliente y la hoja de ruta para la aplicación.

Las aplicaciones educativas además de los aspectos anteriores, deben considerar diversos aspectos, especialmente relacionados con los requerimientos y los temas a tratar en la app, sin embargo a nivel de arquitectura también hay consideraciones

## 11.1 Enfoques de Arquitectura de aplicaciones.

Los enfoques de arquitectura de aplicaciones se pueden clasificar en:

### Aplicación nativa

**Pros:**

- Ofrece la mejor experiencia de usuario, es posible crear aplicaciones complejas, ricas y sensibles que ofrecen el mejor rendimiento.
- Tiene acceso a todas las características nativas proporcionadas por la plataforma.
- Control sobre el almacenamiento en caché de datos local hace que sea posible la implementación de aplicaciones que pueden funcionar fuera de línea.
- Capacidad para garantizar la integridad transaccional en la sincronización de datos fuera de línea.

**Contras:**

- Requiere instalación, actualización y desinstalación.
- Es generalmente muy específica del dispositivo y plataforma.
- Distribución de la aplicación es más complicado y depende a menudo de una tienda.
- Sujeto a la aprobación de la tienda tiempo y podría requerir iteraciones.

### Aplicación basada en Web

**Pros:**

- Reutiliza las aplicaciones web existentes.
- Ajustes menores al CSS y JavaScript son suficientes para hacerlos amigable.
- No requiere instalación, actualización y desinstalación.
- HTML5 pueden resultar ser una ventaja con respecto a algunas de las características nativas como geolocalización, almacenamiento locales, etc
- Puede soportar múltiples dispositivos y versiones de sistemas operativos.
- Posibilidad de crear la interfaz de usuario visualmente atractiva.

**Contras:**

- Rendimiento y facilidad de uso no es tan bueno como aplicaciones nativas.
- Incluso con HTML5, el acceso a la funcionalidad nativa es muy limitada.
- Difícil de implementar aplicaciones que pueden funcionar en un estado desconectado.

### Aplicación híbrida

Aplicaciones híbridas se construyen mediante la combinación de los componentes nativos y componentes web. Componentes Web se construyen utilizando HTML, CSS y JavaScript y mediante un wrapper que da acceso a la funcionalidad nativa a través de JavaScript.

**Pros:**

- Recursos web existentes pueden ser utilizados.
- Acceso a todas las características nativas.
- Puede proporcionar una rica experiencia de usuario mediante el uso de componentes nativos inteligentes

**Contras:**

- Tiene que instalar, actualizar y desinstalar.
- Acceso a funcionalidad nativa utilizando JavaScript viene con una cierta sobrecarga y no es tan eficiente como una aplicación nativa.

La escogencia del enfoques arquitectónico va mucho de la mano de la experiencia y funcionalidad que se le presente al usuario dependiendo de que le que quiera ofrecer al usuario así debe ser el enfoque.

## 11.2 Escogencia del framework de desarrollo.

Luego de seleccionar el enfoque arquitectónico es necesario escoger un framework de desarrollo para esto las siguientes preguntas para determinarlo:

1. ¿Quién va a utilizar la aplicación?
2. ¿Qué tipo de experiencia el usuario espera?
3. ¿Qué funcionalidades son requeridas?
4. ¿Es importante la compatibilidad multiplataforma?
5. ¿Se requiere? Que la aplicación se ejecute fuera de línea
6. ¿Cuánto tiempo se tiene para desarrollar la aplicación?

La conclusión más importante del proyecto es que se puede definir una arquitectura inicial que permita hacer desarrollos de apps para dispositivos móviles. Sin embargo, la gran variedad y posibilidades que brindan los dispositivos, hace necesario caracterizar o clasificar las apps por tipos, para definir arquitecturas específicas para esa caracterización.

La utilización de una metodología definida es de gran importancia en el desarrollo de apps educativas para dispositivos móviles, en este proyecto se implementó en forma exitosa esa metodología para crear dos aplicaciones.

La arquitectura inicial definida es la base para un futuro desarrollo, que permita caracterizar los tipos de aplicaciones móviles educativas para así poder asociar una arquitectura más específica.

# 12. Recomendaciones

Para una continuación del proyecto consideramos pertinentes:

- Realizar una investigación de últimas tendencias en arquitecturas para el desarrollo de apps para móviles: es necesario actualizar la información obtenida al inicio de este proyecto.
- Categorizar las aplicaciones educativas: es necesario definir características generales para segmentos de apps.
- Definición de arquitecturas generales para diversos tipos de aplicaciones móviles educativas.
- Aplicación de las arquitecturas generales en algunos de los diversos tipos de apps educativas.

La recomendación general es que el TEC apoye estos esfuerzos de desarrollo de apps en general y especialmente de apps educativas. Se tienen las capacidades para convertir a la institución en líder en este campo.

# 13. Referencias


1] Chen Y., Kao T., Sheu J. y Chiang Y.. A Mobile Scaftolding-Aid-Based Bird-Watching Learning System. In M.Milrad, H. U. Hoppe and Kinshuk (Eds), IEEEInternational Workshop on Wireless and Mobile Technologies in Education. Los Alamitos, USA: IEEE Computer Society. 2002

[2] Dissanayake Lakshman. mLearning: an innovative conceptualization to expand Education for everyone, anytime, everywhere. Leeds Metropolitan University, UK. Sprouts: Working papers on information systems. ISSN 1535-6078. 2009.

[3] Fallas Ida, Zúñiga Magally. Estudio las tecnologías de la información y las comunicaciones en la educación costarricense. Tercer Informe. Estado de la Educación CONARE. 2010

[4] Goodwin Kristy, Highfield Kate. iTouch and iLearn An examination of educational apps. Early Education and Technology for Children. 2012.

[5] Keith Clinton. Agile game development with Scrum. Pearson Education Inc. 2010. ISBN 0-321-61852-1.

[6] Martín Sergio. Claves para el desarrollo de aplicaciones móviles. IEEE TMC Spain. http://sites.ieee.org/spain-tmc

[7] Millard David E., Faulds Sue J, Gilbert Lester, Howard Ivonne, Sparks Dan, Wills Gary, Zhang Pei. Co-Design for conceptual spaces: an Agile design methodology por m-learning. University of Southampton, UK.

[8] Naismith, L., Lonsdale, P., Vavoula, G., and Sharples, M. Report 11: Literature review in mobile technologies and learning. In Future Series. Nesta Futures Lab, Birmingham, UK. 2005.

[9] Park Yeonjong. A Pedagogical Framework for Mobile Learning: Categorizing Educational Applications of Mobile Technologies into Four Types. The International Review of Research in Open and Distance Learning, Vol 12, No 2. 2011.

[10] Parsons D, Ryu H, Cranshaw M. A Study of Design Requirements for Mobile Learning Environments. Sixth IEEE International Conference on Advanced Learning Technologies (ICALT 2006)

[11] Severin Eugenio, Capota Christine. Modelos uno a uno en América Latina y el Caribe. Banco Interamericano de Desarrollo BID. 2011.

[12] Shepherd Clive. M is for Maybe. http://www.fastrak-consulting.co.uk/tactix/Features/mlearning.htm

[13] UNESCO. Las TIC en la educación. http://www.unesco.org/new/es/unesco/themes/icts/

[14] UNESCO. Mobile Learning and policies: key issues to consider. ISSN-2227-5029. 2012

[15] Valiente Óscar. 1-1 in Education: Current Practice, International Comparative Research Evidence and Policy Implications", OECD Education Working Papers.

[16] Vittone José, Cuello Javier. Diseñando apps para móviles. Primera edición. ISBN 978-84-616-5070-5. Junio 2013.

# 15. Apéndices

## 15.1 Reporte técnico sobre metodologías de desarrollo.

## 15.2 Documentos de metodología definidos.

15.1 Documento de Idea Inicial de Aplicación (IIA)

15.2 Documento de Acta Constitutiva de Proyecto (ACP)

15.3 Documento de Diseño de Aplicación (DDA)

## 15.3 Documento de arquitectura.

## 15.4 Documentos app UBICATEC.

## 15.5 Documentos app PLUS.

# Metodologías en el desarrollo de aplicaciones para dispositivos móviles.

Jeff Schmidt-Peralta

***Abstract*— This technical report issues about mobile app development.**

***Keywords*— metodología, arquitectura, móviles**

## I. Introducción

En el mundo actual los dispositivos móviles son una gran parte de nuestra vida cotidiana, la explosión de la creación de aplicaciones para dispositivos móviles, unido al ritmo de adopción e innovación en dichos dispositivos ha ocasionado que los usuarios se inclinen más por utilizar este tipo de dispositivo en lugar de la tradicional computadora.

Como parte del proyecto de investigación 'Arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles' del Instituto Tecnológico de Costa Rica (TEC), se analizan las diferentes metodologías de desarrollo de aplicaciones para dispositivos móviles. A continuación se muestran los principales resultados del estudio realizado.

## II. Metodologías

Según un informe de International Data Corporation (IDC) el mercado mundial de teléfonos va a alcanzar un total de 1832 millones de unidades incluyendo dispositivos inteligentes para el 2013, resultando en un aumento de 5,5% comparándolo al 2012. Estos altos volúmenes significan una mayor penetración en mercados emergentes así como un mayor alcance a la población mundial.

Todo este crecimiento en el mercado de dispositivos móviles se traduce en más clientes para todo tipo de aplicaciones para dispositivos móviles. Por esta razón es importante determinar cuáles metodologías son más aptas para el desarrollo de aplicaciones en este mercado altamente cambiante y creciente.

Entender las metodologías es importante para poder determinar cuál es la que más se adapta a la forma de desarrollar aplicaciones para dispositivos móviles.

### *A. Metodología cascada*

La ingeniería del software es una disciplina de la ingeniería cuya meta es el desarrollo costeable de sistemas de software. Al ser éste abstracto e intangible, no restringido por materiales ni gobernado por leyes físicas y/o procesos de manufactura; puede ser extremadamente complejo y, por tanto, difícil de entender.

La noción de ingeniería del software fue propuesta inicialmente en 1968 en una conferencia para discutir lo que en ese momento se llamó la crisis del software.

Esta crisis del software fue el resultado de la introducción de las nuevas computadoras hardware basadas en circuitos integrados. Su poder hizo que las aplicaciones hasta ese momento irrealizables fueran una propuesta factible, con órdenes de magnitud grandes y más complejas que sistemas de software previos.

La experiencia previa en la construcción de estos sistemas mostró que un enfoque informal para el desarrollo del software no era muy bueno. Los grandes proyectos a menudo tenían años de retraso. Costaban mucho más de lo presupuestado, eran irrealizables, difíciles de mantener y con un desempeño pobre.

Es por esto que se necesitaron nuevas técnicas y métodos para controlar la complejidad inherente a los sistemas grandes. Estas técnicas han llegado a ser parte de la ingeniería del software y son ampliamente utilizadas.

Ha habido enormes progresos desde 1968; se han comprendido mejor las actividades involucradas en el desarrollo del software y desarrollado métodos efectivos de especificación, diseño e implementación del mismo.

Las nuevas notaciones y herramientas reducen el esfuerzo requerido para producir sistemas grandes y complejos. Sin embargo, no hay un enfoque ideal a la ingeniería del software, pero sí una amplia diversidad de enfoques al desarrollo; esto según la necesidad de cada organización y/o sistema.

Las nociones fundamentales de procesos y la organización del sistema son la base de todas estas técnicas, y éstas son la esencia de la ingeniería del software.

Estos procesos han evolucionado para explotar las capacidades de las personas de una organización, así como las características específicas de los sistemas que se están desarrollando.

Para algunos sistemas, como los sistemas críticos, se requiere un proceso de desarrollo muy estructurado. Para sistemas de negocio, con requerimientos rápidamente cambiantes, un proceso flexible y ágil probablemente sea el más efectivo.

A pesar de que existen muchos procesos diferentes de software, algunas actividades fundamentales son comunes para todos ellos:

1. *Especificación del software*
2. *Diseño e implementación del software*
3. *Validación del software*
4. *Evolución del software*

Algunos modelos de proceso de software generales son:

| Modelo en Cascada |
| --- |
| Considera las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución, y los representa como fases separadas del proceso. |
| **Desarrollo evolutivo** |
| Este enfoque entrelaza las actividades de especificación, desarrollo y validación. Un sistema inicial se desarrolla rápidamente a partir de especificaciones abstractas. Este se refina basándose en las peticiones del cliente para producir un sistema que satisfaga sus necesidades. |
| **Ingeniería del Software basada en Componentes** |
| Este enfoque se basa en la existencia de un número significativo de componentes reutilizables. El proceso de desarrollo del sistema se enfoca en integrar estos componentes en el sistema más que en desarrollarlos desde cero. |

Estos modelos generales no son descripciones definitivas de los procesos del software; más bien son abstracciones de los procesos que se pueden utilizar para explicar diferentes enfoques para el desarrollo. Puede pensarse en ellos como marcos de trabajo del proceso que pueden ser extendidos y adaptados para crear procesos más específicos de ingeniería del software.

Estos tres modelos de procesos genéricos se utilizan ampliamente en la práctica actual de la ingeniería del software. No se excluyen mutuamente y a menudo se utilizan juntos, especialmente para el desarrollo de sistemas grandes. De hecho, el Proceso Unificado de Rational (RUP) que se tratará más adelante, combina elementos de todos estos modelos, así como otros modelos que según el sistema en el que se desarrollen pueden ser estructurados o ágiles.

El primer modelo de proceso de desarrollo de software que se publicó se derivó de ingeniería de sistemas más generales (Royce, 1970). Debido a la cascada de una fase a otra, dicho modelo se conoce como modelo en cascada o como ciclo de vida del software.

Las principales etapas de este modelo se transforman en actividades fundamentales de desarrollo, muy relacionadas a las actividades comunes de la ingeniería del software:

1. **Análisis y definición de requerimientos**

Consiste en la definición de servicios, restricciones y metas del sistema.

2. **Diseño del sistema y del software**

Consiste en dividir los requerimientos en sistemas hardware o software y establecer una arquitectura completa del sistema.

3. **Implementación y prueba de unidades**

Consiste en llevar a cabo el punto anterior y verificar que cada unidad cumpla con su especificación.

4. **Integración y prueba del sistema.**

Consiste en integrar y probar las unidades individuales y programas, esto como un sistema completo, con el fin de asegurar los requerimientos del software para el cliente.

5. **Funcionamiento y mantenimiento.**

Consiste en instalar y poner en funcionamiento práctico el sistema. Además de corregir errores no descubiertos en etapas anteriores del ciclo de vida, mejorar la implementación y resaltar los servicios ante nuevos requerimientos.

Davis et al. (1988) citó los siguientes usos del modelo en cascada:

1. El modelo fomenta especificar lo que el sistema debe hacer antes de construir o desarrollar el sistema.
2. Fomenta cómo los componentes van a interactuar entre sí.
3. Permite a los administradores de proyecto seguir el proyecto con más precisión y descubrir posibles amenazas a tiempo.
4. Demanda que el proceso de desarrollo genere una serie de documentos que pueden ser utilizados más tarde para pruebas y mantenimiento del sistema. Además de que esta documentación generada en cada fase cuadra con otros modelos de proceso de ingeniería.
5. Reduce los costos de desarrollo y mantenimiento.
6. Permite desarrollar sistemas más estructurados y administrados.

Aunque el modelo provee un práctico y disciplinado enfoque para el desarrollo del software, cuenta con los siguientes problemas (Royce, 2000):

1. **Integración Prolongada y División de Diseño Tardío.**

Fuerte énfasis en el análisis y el diseño perfecto a menudo resultaba en demasiadas reuniones, y documentación en exceso, y sustancialmente retrasando el proceso de integración y pruebas, sin óptimas soluciones, muy poco tiempo para el rediseño, y con retraso en la entrega de productos no mantenibles.

2. **Manejo de la Descomposición Funcional de Requerimientos.**

El modelo de cascada requiere especificación de requerimientos completos y sin ambigüedades. Pero también supone que todos los requisitos son igualmente importantes, y que no cambian a lo largo de las fases de desarrollo.

Otras críticas a las que ha estado expuesto el modelo son las siguientes:

- **El modelo de cascada es rígido**: La rigidez en sus fases es debido a que si una fase aún no ha terminado su proceso de 'vida', no se pueden inicializar con la siguiente etapa.

En principio, el resultado de cada fase es uno o más documentos aprobados. En la práctica, estas etapas se superponen y proporcionan información a otras.

- **Es monolítico:** La planeación es orientada a una única fecha de entrega. Si un error ocurre en la etapa de análisis, esto va a ser conocido por el usuario final cuando se entregue el producto total.

- **El modelo está fuertemente basado en documentación,** hasta el punto de ser burocrático.

- **El proceso del software no es un modelo lineal simple**: Esto debido a que implica una serie de iteraciones de las actividades de desarrollo.

Debido a los costos de producción y aprobación de documentos, las iteraciones son costosas e implican rehacer el trabajo. Por lo tanto, después de un número reducido de iteraciones, es normal congelar partes del desarrollo, como la especificación, y continuar con las siguientes etapas de desarrollo.

Este congelamiento prematuro de requerimientos puede implicar que el sistema no haga lo que los usuarios desean. También puede conducir a sistemas mal estructurados debido a que los problemas de diseño se resuelven mediante trucos de implementación además, los problemas se posponen para su resolución, se pasan por alto o se programan.

Además en el caso de que los requerimientos del usuario no estén lo suficientemente claros o dichos requerimientos cambien durante el diseño, codificación y etapas de testeo, el modelo de cascada resulta ser inadecuado.

Esto debido a que su principal problema es su inflexibilidad a dividir el proyecto en etapas distintas. Se deben hacer compromisos en las etapas iniciales, lo que hace difícil responder a los cambios en los requerimientos del cliente.

Por lo tanto, el modelo Cascada sólo se debe utilizar cuando los requerimientos se comprendan bien y sea improbable que cambien radicalmente durante el desarrollo del sistema.

### *A.A.1 Modelo Cascada y desarrollo de aplicaciones móviles.*

No se encontraron referencias en relación a la aplicación del Modelo Cascada en el desarrollo de aplicaciones móviles.

Algunas de las posibles explicaciones o respuestas a esta interrogante, es la actual implementación de Metodologías Ágiles de desarrollo.

Debido a las problemáticas y críticas dadas en el Modelo Cascada (citados en el apartado anterior), las metodologías ágiles de desarrollo están especialmente indicadas en proyectos con requisitos poco definidos o cambiantes.

Estas metodologías se aplican bien en equipos pequeños que resuelven problemas concretos, lo que no está reñido con su aplicación en el desarrollo de grandes sistemas, ya que una correcta modularización de los mismos es fundamental para su exitosa implantación.

## *B. Metodología RUP*

La metodología RUP o Rational Unified Process por sus siglas en inglés, es una serie de procesos que fueron desarrollados por la empresa Rational Software, que en la actualidad es propiedad de IBM. Junto con el Lenguaje Unificado de Modelado (UML), constituyen la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos.

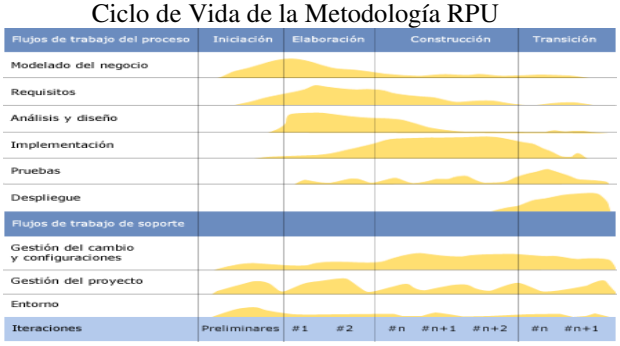
RUP es una metodología que posee un enfoque específico para asignar tareas y responsabilidades dentro de una organización de desarrollo. Posee como objetivo principal asegurar la producción de software de alta calidad dentro de un tiempo y presupuesto previsibles. Es un metodología de desarrollo iterativo enfocada hacia los casos de uso, manejo de riesgos y manejo de la arquitectura.

El RUP pretende mejorar la productividad del equipo de trabajo, ya que permite a cada miembro del equipo sin importar su responsabilidad acceder a la misma base de conocimientos. Lo cual obliga a todos a compartir el mismo lenguaje, la misma visión y el mismo proceso acerca de cómo desarrollar software.

Como se puede observar en la Tabla I, el ciclo de vida RUP es una implementación de desarrollo en espiral, se establecen tareas en fases e iteraciones. El RUP posee cuatro fases, dentro de las cuales se realizan iteraciones en números variables.

Las primeras iteraciones, en la fase de inicio y elaboración, son dirigidas a la comprensión del problema y la tecnología, la delimitación del ámbito del proyecto, eliminación de riesgos y un establecimiento de unas buenas bases para el inicio.

TABLA I
Ciclo de Vida de la Metodología RPU



*B.A.1 Fases*

*B.A.1.1 Fase de Inicio*

Durante esta fase las iteraciones que se realizan ponen un mayor énfasis en las actividades modelando el negocio y requisitos.

- Modelando el negocio: Durante esta fase el equipo de trabajo se familiariza entre sí y con la empresa, para conocer sus políticas su forma de trabajo. A demás de asegurarse de que la organización y los desarrolladores tengan claro el objetivo del proyecto.
- Requisitos: Son los lineamientos del proyecto, por lo cual los usuarios finales deben comprender y aceptar todos los requisitos. Se debe establecer y mantener un acuerdo entre los clientes y los stakeholders sobre los alcances del proyecto.

*B.A.1.2 Fase de Elaboración*

En esta fase todas las iteraciones son orientadas al desarrollo de la base de la arquitectura, se revisa y corrige el modelo de negocios, se centra mas en la etapa de análisis y diseño del sistema y un poco en la implementación. Se especifican los requerimientos, se da una descripción sobre cómo se va a implementar el sistema. Se transforman los requisitos en diseños del sistema, se desarrolla la arquitectura.

*B.A.1.3 Fase de Construcción*

En esta fase se inicia con lo que es la construcción del sistema.

- Implementación: Se implementan todas las clases y objetos en ficheros fuente, ejecutables, entre otros. La salida de esta implementación es un sistema ejecutable. A demás se planifica que más subsistemas deben ser implementados, y en qué orden deben ser ensamblados estos diferentes módulos. En caso de encontrar errores en el sistema, estos deben ser notificados.
- Pruebas: En las parte de pruebas, el encargado evalúa la calidad de producto que se está desarrollando, pero no para aceptar o rechazar lo que se está realizando, más bien es para integrarlo en el ciclo de vida. En esta parte se encuentran y documentan fallas de software.

*B.A.1.4 Fase de Transición*

Despliegue: El fin de esta etapa es el de producir con éxito el sistema, además de distribuirlo a los usuarios finales. Se prueba el producto en su entorno de ejecución final, se empaqueta el software para su distribución. Se instala el software (en caso de ser necesario) al usuario final. Se debe capacitar a los usuarios en el cómo utilizar el software.

***Ventajas***

Una de las principales ventajas es que durante todo el proyecto, este se debe gestionar. Esto significa que se debe vigilar el cumplimiento de los objetivos la buena gestión de los riesgos y las restricciones para el buen desarrollo del producto, que este cumpla con las expectativas del usuario, lo cual genera una retroalimentación muy temprana en el desarrollo. Por otro lado se encuentra la documentación que es uno de los puntos más importantes de la metodología RUP, ya que todos los cambios que se realicen todos los procesos deben de documentarse lo cual es ventajoso para poder llevar un orden.

Esta metodología se base en mejorar prácticas que la empresa posee, aplicarlas de mejor forma para obtener mejores resultados, como mitigación temprana de riesgos, temprana retroalimentación, entre otras.

***Tendencias***

Las tendencias que se encuentran en la actualidad en cuanto al desarrollo de aplicaciones móviles, se da por el lado de lo que se conoce como metodologías agiles, la metodología RUP no se encuentra entre estas denominadas metodologías agiles, pero todavía es utilizada por algunas empresas para el desarrollo de aplicaciones móviles, debido a su orden para realizar las actividades.

TABLA II
Tabla de Ventajas

| | |
|---|---|
| Mitigación Temprana de Riesgos | Como todo se controla de forma tan rápida cuando un riesgo se estima se puede presentar se toman todas las precauciones necesarias |
| Visible progreso en las primeras etapas | En las primeras etapas si bien es cierto es mucha documentación, esta sirve para que los stakeholders vean un avance rápido en las primeras etapas |
| Temprana retroalimentación | Ya que con la documentación que se debe de realizar todo es documentado lo cual sirve para que los involucrados lo revisen y den su opinión para usar esta en forma de feedback. |
| Conocimientos reutilizable | Todo conocimiento que se recompile en las primeras etapas del proyecto, puede ser reutilizado en las demás etapas del proyecto |
| Alta efectividad en proyectos a largo plazo | Esta metodología se adapta mejor a proyectos que son a largo plazo, ya que por su cantidad de documentación su duración se vuelve muy extensa. |

*C. Metodología AUP*

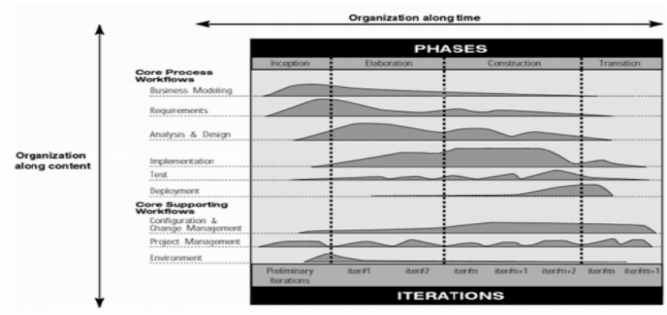
La metodología AUP o Agile Unified Process por sus siglas en inglés, es un enfoque metodológico ágil basado en RUP. Es un enfoque simple y fácil de entender para el desarrollo de software, usando técnicas y conceptos procedentes del RUP.

AUP es una metodología que fue creada por Scott Amber, un ingeniero dedicado a trabajo e investigación de mecanismos agiles para el desarrollo de software. En el 2005 libero la primera versión del AUP en un sitio WEB

desarrollado específicamente para ayudar a desarrollar esta metodología.

Al igual que la metodología RUP, que se encuentra formada por fases, la metodología AUP también posee fases, posee cuatro, las cuales son fase de inicio, fase de elaboración, fase de desarrollo y fase de transición.

TABLA III
CICLO DE VIDA DE LA METODOLOGÍA RPU




### *C.A.1 Fases*

#### *C.A.1.1 Fase de inicio*

Durante esta fase lo que mayormente se realiza es documentación, la cual a lo largo del desarrollo se debe ir actualizando y modificando, dependiendo de cómo se vaya desarrollando el proyecto. La documentación que debe quedar lista en esta primera etapa es:

- El alcance del proyecto: Llegar a un acuerdo con todas las personas involucradas en el desarrollo de la aplicación, de cuál va a ser el alcance final del software.
- Definición inicial de los requerimientos.
- Documentación de los riesgos que se espera puedan aparecer a lo largo del desarrollo de la aplicación y uno o más planes de contingencia para cada situación.

Dejar listo un plan del proyecto, calendarizar todos los procesos que se deben de realizar para asegurarse que la aplicación va a estar lista en el tiempo planeado.

#### *C.A.1.2 Fase de Elaboración*

En esta fase se revisa que exista una estabilidad es la visión propuesta en la fase anterior, se empieza a desarrollar la arquitectura de la aplicación. A demás se da una revisión de los riesgos, para observar si los propuestos anteriormente aún son latentes, si existen nuevos, y se revisan los diferentes planes de contingencia. Se examina el plan del proyecto para ver si este es todavía viable, si todo se va realizando acorde a lo planeado.

#### *C.A.1.3 Fase de Construcción*

En esta fase se inicia con lo que es la construcción del sistema. Se debe procurar que el sistema sea lo más estable posible, se debe ir preparando a los demás involucrados, ya sea con capacitación de cómo usar el software, o en caso de aplicaciones móviles se debe empezar a desarrollar el manual de usuario y versiones beta. Se da una revisión más de los riesgos, para saber si si estos aún pueden afectar y si los planes vigentes para contenerlos son aun efectivos, y también para verificar la existencia de nuevos riesgos. Se empiezan a estimar los costos de la aplicación, además se actualiza el plan del proyecto para saber cuánto tiempo queda disponibles para los procesos que aún no se terminan.

#### *C.A.1.4 Fase de Transición*

En esta fase, la fase final, se busca una aceptación de los clientes, o del público en general de la aplicación realizada, se estima el coste total de la aplicación, se debe brindar soporte a los clientes hasta que estos sean capaces de dar un mantenimiento a la aplicación. Se liberar los manuales de usuarios y las versiones beta para los usuarios de dispositivos móviles.

***Ventajas***

Durante todo el proyecto, este se debe gestionar. Esto significa que se debe vigilar el cumplimiento de los objetivos la buena gestión de los riesgos y las restricciones para el buen desarrollo de la aplicación, que este cumpla con las expectativas del usuario, lo cual asegura una retroalimentación temprana en el desarrollo de la aplicación. A demás la ventaja que posee AUP sobre RUP es que AUP es una versión simplificada de la metodología RUP, con lo cual se posee toda la efectividad de la metodología RUP pero en procesos más que mejoran la efectividad en tiempo del desarrollo de la aplicación. Puede utilizarse y adaptarse al desarrollo de aplicaciones móviles, ya que al ser un derivado de RUP esta se vuelve más ligera, si bien tiene la misma cantidad de fases, posee menos procesos por lo cual permite que el software se realice en menor tiempo debido a que no hay que llenar tantos documentos, pero si se llevan documentos los cuales permiten llevar un control adecuado del proyecto

***Tendencias***

Las tendencias que se encuentran en la actualidad en cuanto al desarrollo de aplicaciones móviles, se da por el lado de lo que se conoce como metodologías agiles, la metodología AUP se encuentra entre estas denominadas metodologías agiles, por lo que esta metodología es considerada como una excelente opción para el desarrollo de aplicaciones móviles.

TABLA IV
CICLO DE VIDA DE LA METODOLOGÍA RPU

| | |
|---|---|
| Detección temprana de riesgos | Gracias a la rápida retroalimentación que existe |
| Administración adecuada del cambio | Gracias a la documentación todo los cambios son documentados lo cual permite que se manejen de la mejor manera |
| Mayor grado de reutilización | Al igual que la metodología RUP toda información obtenida en etapas previas puede ser utilizada en las futuras etapas del desarrollo |

TABLA V
Entregables de Metodología AUP

| Entregables | Descripción |
|---|---|
| Sistema | Software |
| Código Fuente | Código de todo el programa. |
| Suite de Pruebas de | Casos de pruebas, pruebas de sistema |
| Scripts de Instalación | |
| Documentación del sistema | Documentación, Manuales, entre otros. |
| Notas de Versiones | Control de versiones |
| Modelado de Requerimientos | Descripción de lo que debe de hacer el sistema. Casos de Uso, Requerimientos, Reglas de negocio, Modelo de dominio. |
| Modelo de Diseño | Diseño del sistema. |

*D. Metodologías Agiles*

El método de desarrollo ágil, se basa en un proceso repetitivo donde los problemas y resultados evolucionan constantemente. Como explica Holcombe [1] el proceso de desarrollo ágil de aplicaciones tiene que lograr adaptarse de manera rápida a los cambios, sin perder la calidad y seguridad de la aplicación de manera que el costo-beneficio sea el mejor.

Holcombe[1] menciona que para que un método ágil se considere como tal, tiene que tener ciertas características básicas como las siguientes.

Lograr adaptarse a cambios en las especificaciones de manera natural, pero a la vez de una manera segura, sin que esto represente un gran costo de tiempo, o de recursos. Una manera de lograrlo es anticipando posibles cambios.

Trabajar de manera modular los elementos de la aplicación, permite realizar cambios sólo a los módulos involucrados directamente en los cambios.

Inclusive se tienen los principios que deben seguir las metodologías ágiles. Los siguientes son algunos ejemplos.

- "Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor".
- "Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente."
- "Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible."
- "La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial".
- "Las mejores arquitecturas, requisitos y diseños emergen de equipos auto-organizados".

Koch[2] menciona que en grupos de trabajo, la cultura organizacional influye en la manera en que los problemas son resueltos. Por ejemplo quien tiene la autoridad de realizar cambios, o cual es el procedimiento a seguir para realizar uno. Sin alguno de estos aspectos la aplicación podría tener errores debido a la falta de comunicación o documentación al realizar cambios en el grupo. Por lo cual es de vital importancia utilizar algún procedimiento que comunique a todo el grupo para realizar un cambio.

"Cuando iniciamos un nuevo proyecto de desarrollo de software, lo inicial y algunos dirían la fase más dura, es determinar los requerimientos del proyecto, saber qué es lo que el cliente quiere." Holcombe [1]. Sin embargo, en las metodologías ágiles se aceptan los cambios y se corrigen estos requerimientos. La esencia de lo ágil, es poder modificar "lo que se quiere" mientras se desarrolla.

*D.A.1 Scrum*

Scrum no es exactamente una metodología ágil, es un marco de trabajo para mejorar la efectividad de los equipos en proyectos de software. Busca lograr calidad, agilidad en los cambios y rapidez a la hora de completar el proyecto. Requiere de una comunicación continua con el cliente, por ejemplo, pequeñas reuniones todas las semanas. Scrum busca Existen tres roles principales en Scrum:

- Dueño del Producto: se encarga de priorizar tareas. ¿Qué es lo más importante en los próximos 30 días?
- Equipo de desarrollo: se encarga de realizar el producto y mostrar los avances al dueño del producto para que éste pueda determinar que se desarrollará ahora.
- Maestro Scrum: Se encarga de agilizar el proceso y verificar que se siga bajo el marco de trabajo Scrum.

La característica principal de Scrum es el Sprint. Un Sprint es una unidad de tiempo menor a un mes en la que se realiza un incremento, terminado, funcional y potencialmente entregable al cliente. El equipo de desarrollo es quién decide las funcionalidades que se van a desarrollar durante éste y además estiman la duración del sprint. Se realizan una reunión de planificación del sprint al inicio de éste, reuniones diarias llamadas Daily Scrums donde los miembros del equipo de desarrollo indican que han logrado, que realizarán durante el día y que problemas u obstáculos enfrentan. También se realizan la revisión del sprint, y la retrospectiva del sprint. Durante un sprint no se deben realizar cambios que afecten el objetivo principal del sprint, cambiar el equipo de desarrollo, disminuir la calidad del incremento, pero sí se puede negociar el alcance entre el equipo de desarrollo y el dueño del producto.

La pila de producto, o product backlog, es una lista ordenada con todo lo que podría ser utilizado en el producto, se trata de los requerimientos del producto escritos en lenguaje humano y entendible por el cliente. Nunca está completa ya que evoluciona conforme el producto se va desarrollando. Se ordena normalmente por valor, riesgo,

prioridad y necesidad. La pila del sprint, es una lista con los requerimientos que se desarrollarán durante el sprint, más un plan para entregar el incremento y conseguir el objetivo.

Scrum lo que busca es dar valor agregado al producto, mientras aumenta la calidad del mismo mediante el crecimiento personal de los miembros del equipo de trabajo. El equipo adquiere responsabilidad y compromiso, y el cliente se ve envuelto casi como un miembro del equipo de trabajo.

*D.A.2 Lean*

Iniciado por Mary y Tom Poppendieck en el libro "Lean Software Development", nace Lean cómo el precursor de las metodologías ágiles. En este libro, se presentan los principios de esta metodología y una serie de 22 herramientas. La metodología se resume en siete principios que dan ventajas sobre otras metodologías.

1. Eliminar el desperdicio: Todo lo que no le agregue valor al cliente es desperdicio. Esto incluye:

Código y funcionalidades innecesarias.
Atraso en el desarrollo.
Burocracia.
Pocas pruebas.

Amplificar el aprendizaje: Esto es para crear conocimiento en el equipo de desarrollo. La mejor manera de lograr cada vez mejor software es mejorando la calidad de los desarrolladores. Además, incrementar la comunicación entre el cliente y los desarrolladores para que ambas partes entiendan mejor lo que quieren.

Posponer las decisiones: Entre más tarde se decida, mejor. Esto debido a los cambios que ocurren en el proceso de desarrollo de software. Entre más avanzado esté el proyecto los clientes van a saber mejor cuáles son sus necesidades y van a poder tomar las mejores decisiones.

Construir la calidad desde adentro: Existen procesos, y buenas prácticas, diseñados para evitar problemas de calidad. El cliente necesita además, ver el sistema como un todo, así que se debe integrar el sistema para que en cada iteración el cliente observe las mejoras.

Entregar rápido: Al cliente le gusta observar el avance. Entre más rápido y con menos defectos se entregue el cliente va a poder brindar mayor retroalimentación al equipo.

Respetar a las personas: Parece lógico, pero ayuda en el avance y la cooperación. Los líderes de la organización deben escuchar a los desarrolladores para entender cuáles acciones deben ser tomadas y lograr seguir una guía para mejorar el proceso.

Optimizar todo: Responder rápido, escuchar con atención, tomar en cuenta. No se deben posponer los defectos, deben atenderse con forme aparezcan.

*D.A.3 Extreme programing*

Extreme Programming hace hincapié en el trabajo en equipo. Los gerentes, clientes y desarrolladores son socios iguales en un equipo de colaboración. Extreme Programming implementa un entorno simple, pero eficaz permite a los equipos a ser altamente productivos.

La programación extrema se puede definir como un proceso de desarrollo de software para ser utilizado para integración de equipos y para situaciones las cuales el cliente establezca información simple sobre aspectos funcionales y no funcionales para su aplicación. Los aspectos de la programación extrema se complementan para la generación del proyecto, además que requiere del apoyo de los miembros internos y externos.

Consiste basicamente en ajustarse estrictamente a una serie de reglas que se centran en las necesidades del cliente para lo grar un producto de buena calidad en poco tiempo.

La Programación Extrema es una metodología ágil centrada en potenciar las relaciones interpesonales como clave para el éxito en el desarrollo de software.Promueve el trabajo en equi po, preocupandose en todo momento del aprendizaje de los d esarrolladores y estableciendo un buen clima de trabajo.

Este tipo de método se basa en una realimentación continua da entre el cliente y el equipo de desarrollo con una comunica ción entre todos los participantes, también busca simplif icar las
soluciones implementadas y coraje para los múltiples cambio s.

Esta metodología busca que tanto programadores como cliente y usuarios finales estén en constante comunicación, además para los equipos de desarrollo busca que estos aprendan nuevas destrezas y complementen sus habilidades

Parla la programación extrema se busca que los miembros estén definidos y con roles claros, los roles que se mostraran a continuación buscan aclarar que miembros son los que involucra un proyecto de esta perspectiva.

*D.A.3.1 Historias del Usuario*

Esta es la técnica base para la metodología Extreme Programming, ya que es la técnica usada en la recopilación de los requisitos de software.

Las historias de usuario son en sí mismo una historia la cual detalla que se desea colocar en el proyecto que piensa el cliente y los usuarios finales respecto a que debe hacer la aplicación, también mencionan puntos de referencia sobre posibles aplicaciones pasadas que no cumplían con todos los requisitos o que fueron insatisfactorias en los resultados todo basado en las experiencias de los usuarios, también se pueden colocar ideas de los usuario respecto a algún aspecto de comodidad en la aplicación la cual puede basarse en aplicaciones que ellos manejes en la empresa.

Este proceso es simple, pero cada historia de usuario debe ser comprensible para que las ideas sean usadas por los programadores en el desarrollo de la aplicación.

Las historias de usuario cambian de acuerdo a los equipos de trabajo pero debe contener al menos los siguientes elementos: fecha, tipo de actividad (nueva, mejora, corrección), prueba funcional, número de historia, prioridad técnica y del cliente, referencia de historias previas, riesgo,

estimación técnica, descripción, notas, lita de seguimiento con fecha, estado, lista de pendientes y comentarios.

Para efectos de planificación de los proyectos bajo esta metodología las historias de usuario pueden generarse una cada tras semanas, las historias pueden transformarse en tareas de programación después de ser descompuestas en la reunión con el cliente, al final se les asigna a los programadores las diferentes tareas para ser implementadas en una iteración.

*D.A.3.2 Estudios de Usuario*

Los estudios de usuario son la fuente de conocimiento acerca de los usuarios finales.

Este proceso se implementa de dos diferentes maneras:

a) Entrevistas: Esta es una técnica que es muy utilizada en el marketing y la psicología. Actualmente es también muy utilizada para la recopilación de información de los usuarios finales.
b) Campos de pruebas: En campos de prueba, el producto o funcionalidad que ya haya sido desarrollada es puesta a prueba en un entorno no controlado. La información que pueda generar esta prueba, puede ser recabada, mediante diferentes técnicas, como están, los cuestionarios, entrevistas o las observaciones diarias.

*D.A.3.3 Procesos de Extreme Programming*

Para un proyecto de Extreme Programming se deben seguir los siguientes procesos:

1. El cliente define el valor del negocio respecto al proyecto. (El valor del proyecto para la empresa del cliente)

2. Se realiza la estimación de los esfuerzos que requieren equipos técnicos. (Diseñadores programadores técnicos y especialistas)

3. El cliente define qué actividades se ejecutan primero basados en las restricciones de tiempo presupuesto y prioridad del cliente.

4. El equipo técnico genera ese valor de negocio.

Pruebas comentarios y regreso al primer pasó hasta terminar con el proyecto.

*D.A.3.4 Prácticas en Extreme Programming*

**Planificación:** El equipo técnico en conjunto con el cliente definen los plazos, el esfuerzo y las entregas de producto basados en las experiencias de usuario que proporciona el cliente.

**Entregas:** Son entregas de partes del proyecto que pueden presentarse ante el cliente aun cuando estas no son el producto final facilita la revisión del proyecto, estas entregas no deben tardar más de 3 meses.

**Metáfora:** Este se puede definir como una historia conjunta entre equipo técnico y cliente para poder llegar a puntos en común de cómo deberá funcionar el sistema mediante el uso de un vocabulario que ayude a la resolución de los métodos de desarrollo del sistema.

**Diseño simple**: Es generar el diseño más simple que se pueda respecto a la aplicación para poder ser usado en la misma en un momento determinado.

**Pruebas:** Son las pruebas que el cliente solicita que sean realizadas, estas solicitudes se efectúan antes y después de generar el código y se realizan cada vez que se genera una actualización.

**Programación en Parejas:** Este punto establece que todo aspecto de programación debe de realizarse en parejas lo que permite menor número de errores, mejor diseño y la propiedad colectiva de código lo que permite a ambos programadores realizar cambios al código.La pareja cambia los papeles con frecuencia durante el día.

**Integración continúa:** Este aspecto consiste en integrar cada elemento del código al proyecto en el momento de que este esté funcionando correctamente. Así se pueden realizar varias integraciones por día.

**Diseño Simple:** Es asumir que los requisitos cambian constantemente, el futuro es incierto, y los costes de cambio no son exponenciales. Por lo tanto, el desarrollo más rentable debe centrarse en resolver los problemas de hoy, en lugar de diseñar para futuros cambios. En extreme programming, el mejor diseño de software se ejecuta en todos los pilotos de pruebas, no tiene despidos de código, tiene el menor número de clases posibles y métodos, y es fácil de entender.

La motivación detrás de un diseño simple es que, en vista de extreme programming, los requisitos no son completos cuando el diseño comienza. Esto está en consonancia con la realidad de las aplicaciones móviles, que tienen que responder a las presiones del mercado y de la competencia que están fuera de su control, u otras circunstancias inevitables, tales como las variaciones en la tecnología de implementación.

Por lo tanto, el diseño es mínimo basándose en una corriente de requisitos. Se apuesta por la simplicidad, y para garantizar el "buen" diseño su calidad (específicamente, la complejidad

estructural) está mejorarse revisiones frecuentes, es decir, refactorización.

**40 horas por semana:** Se debe trabajar máximo 40 horas por semana y no se trabajaran horas extra por dos semanas seguidas, si estos objetivos no se cumplen de alguna manera se está incurriendo en una falla la cual debe ser corregida, esto porque el trabajo extra desmotiva a los equipos.

**Cliente in-situ:** Este punto se enfoca en que el cliente siempre está presente o disponible para el equipo desarrollo, esto es vital para el éxito en un proyecto que utiliza extreme programming.
El cliente con esto conduce el proyecto por la dirección que genere mayor valor al negocio, además los programadores están disponibles para resolver cualquier duda asociada

*D.A.3.5 Pruebas de Usabilidad*

En este tipo de procedimiento, al igual que en el caso anterior, se le proporciona al usuario la aplicación desarrollada, para que este proceda a utilizarla.

En este caso se realiza una observación constante durante todo el proceso. En algunas ocasiones se le pregunta al usuario sobre su experiencia, para que basado en sus pensamientos, poder intuir el modelo mental del usuario y así poder obtener posibles problemas que se puedan encontrar en el sistema.

*D.A.3.6 Evaluación de Expertos sobre Usabilidad de Usuario*

Este se refiere al análisis de expertos acerca de las conductas de los usuarios ante el uso de determinada aplicación.
Existen de dos tipos, las cuales son:

a) *Evaluaciones heurísticas:* Este método de evaluación es muy considerado ya que posee un eficiente análisis y es de bajo costo. Además puede ser repetido constantemente durante todo el proceso de desarrollo.
   Durante la evaluación heurística de tres a cinco expertos evalúan el sistema, acorde con una lista de heurísticas o criterios. Los posibles problemas que son encontrados, son categorizados, dependiendo de su grado se severidad. Sin embargo, este tipo de proceso de evaluación no cubre todos los posibles problemas de usuario.
b) *Tutorial cognitivo:* El objetivo principal de este método de análisis, es el de medir el grado de facilidad de aprendizaje al detectar problemas de usuario.

*D.A.3.7 Juego de Planificación*

El propósito del juego de planificación es determinar el alcance de las emisiones del proyecto y futuras mediante la combinación de las prioridades y las estimaciones técnicas. Para ello, se solicita información desde el "cliente" para definir el valor deseadoen la aplicación, características y usos de las estimaciones de costos proporcionada por los programadores.

Esta entrada se presenta en forma de historias de usuario. La estimación se limita a la evaluación de la velocidad de proyecto, un tangible métrico que determina el ritmo al que el equipo puede producir entregables. El plan es propenso a modificaciones basadas en la realidad actual.

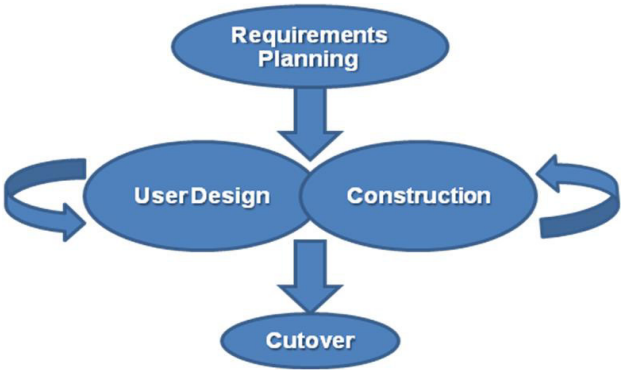
E. *RAD*

El proceso de desarrollo rápido de aplicaciones por sus siglas en ingles RAD, es una variable de las metodologías de desarrollo ágil, esta metodología permite construir sistemas utilizables en poco tiempo normalmente de 60 a 90 días con algunas concesiones. Fue desarrollado inicialmente por James Maslow en 1980. El método comprende el desarrollo interactivo, la construcción de prototipos y el uso de utilidades.

***Principios de la Metodología RAD***

Los principios tras el RAD se basan esencialmente en la posibilidad de crear una aplicación utilizable en un 80% en el 20% del tiempo requerido para la solución completa, el RAD hace énfasis en la posibilidad de que los requisitos de negocio de un sistema puede satisfacerse aun cuando algunos requisitos operacionales no se satisfagan ,por último el RAD detalla que la aceptabilidad de un sistema puede determinarse en base a un conjunto mínimo de requisitos consensados en lugar de la totalidad de los requisitos.

CUADRO II
MODELO DESARROLLO RAD



*Problemas atendidos por el (RAD)*

- Con los métodos convencionales pasa un gran lapso de tiempo antes de que el cliente vea resultados.
- Con los métodos convencionales el desarrollo llega a tardar tanto que para cuando el sistema está listo para utilizarse los procesos del cliente han cambiado radicalmente.
- Con los métodos convencionales no hay nada hasta que el 100% del proceso de desarrollo se ha realizado, entonces se entrega el 100% del software.

## CARACTERÍSTICAS DEL RAD

### A. *Equipos Híbridos*

- Equipos compuestos por alrededor de seis personas, incluyendo desarrolladores y usuarios de tiempo completo del sistema así como aquellas personas involucradas con los requisitos.
- Los desarrolladores de RAD deben ser "renacentistas": analistas, diseñadores y programadores en uno.

### B. *Herramientas Especializadas*

- Desarrollo "visual"
- Creación de prototipos falsos (simulación pura)
- Creación de prototipos funcionales
- Múltiples lenguajes
- Calendario grupal
- Herramientas colaborativas y de trabajo en equipo
- Componentes reusables
- Interfaces estándares (API)
- Control de versiones

### C. *Timeboxing*

Las funciones secundarias son eliminadas como sea necesario para cumplir con el calendario.

### D. *Prototipos Iterativos y Evolucionarios*

*Reunión JAD (Joint Application Development):*

- Se reúnen los usuarios finales y los desarrolladores.
- Lluvia de ideas para obtener un borrador inicial de los requisitos.

*Iterar hasta acabar:*

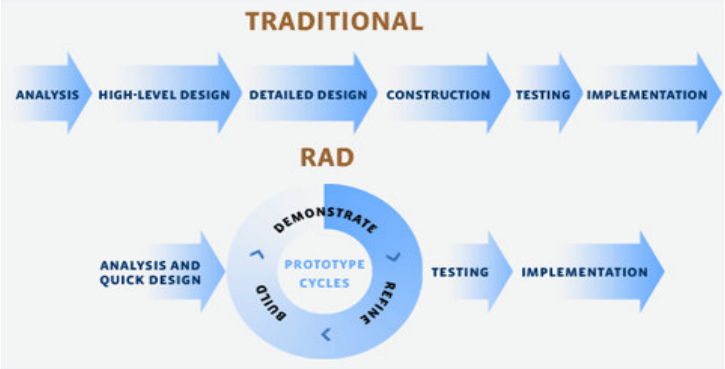
- Los desarrolladores construyen y depuran el prototipo basado en los requisitos actuales.
- Los diseñadores revisan el prototipo.
- Los clientes prueban el prototipo, depuran los requisitos.
- Los clientes y desarrolladores se reúnen para revisar juntos el producto, refinar los requisitos y generar solicitudes de cambios.
- Los cambios para los que no hay tiempo no se realizan. Los requisitos secundarios se eliminan si es necesario para cumplir el calendario.

### E. *Notas:*

- Cada iteración dura entre un día y tres semanas.
- Reuniones de 2 horas con facilitador que mantiene enfocado al grupo.

CUADRO IIII

COMPARATIVA DE METODOLOGÍA RAD CON LOS METODOLOGÍAS TRADICIONALES




## VENTAJAS DEL RAD

Una de las principales ventajas de la implementación de la metodología RAD es que esta permite visualizar de manera temprana resultados del desarrollo de sistema, esto permite atender a tiempo las posibles fallas del sistema antes de que se hayan invertido gran cantidad de tiempo y recursos por lo cual posiblemente reduzcan los costos, otro aspecto que cabe

resaltan dentro del RAD una considerable reducción del tiempo esto es posible debido a que en primera instancia el equipo de trabajo se mantiene enfocado durante el desarrollo sistema permitiendo con esto cumplir con el cronograma planteado.

## III. DISCUSIÓN.

Actualmente los expertos apuntan a usar las metodologías ágiles. Según ellos, estos modelos de trabajo siguen esquemas poco rígidos, por lo que hacer un cambio no es una ardua tarea; algo a lo que están acostumbrados puesto que cuentan que muchas veces durante el desarrollo, a los clientes les surge una nueva idea o una nueva funcionalidad que la aplicación debe tener, lo que implica cambios en el diseño original. Por fortuna, estas metodologías permiten que los cambios en los requerimientos sean fáciles de aplicar, así cumpliendo con las necesidades del cliente sin la obligación de diseñar nuevamente las cosas o ejecutar ciertos procedimientos que se tienen que llevar a cabo en otras metodologías con esquemas más fuertes.

Hooman Bahador de la empresa Konrad Group, indicó que ellos utilizan una metodología ágil debido a que es la más apta para este ambiente ya que el desarrollo de aplicaciones para dispositivos móviles es muy cambiante y es esencial para ellos adaptarse rápidamente a los cambios de plataformas, dispositivos, frameworks, APIs y requerimientos.

En lo que diseño concierne, los expertos fueron muy claros con que una aplicación debe ser visualmente atractiva y fácil de utilizar. Ellos señalan que una aplicación que no sea intuitiva y que al usuario le cueste hacer lo que necesita hacer; rápidamente será tachada como mala, a pesar de que su funcionamiento interno esté muy bien hecho. Conste que esto no significa que se debe descuidar el diseño de la funcionalidad para enfocarse en el diseño visual de la aplicación, sino que se le debe dar un cuidado equitativo a ambas partes puesto que el usuario también necesita que la aplicación funcione de manera eficiente y eficaz. Para lograr una usabilidad eficaz, los expertos recomendaron consultar las guías que Apple y Android han publicado sobre cómo debe ser la usabilidad de una aplicación para cada plataforma.

A pesar de cumplir con los requisitos de usabilidad y de diseño una aplicación buena que no tenga publicidad difícilmente será llegada a conocer por las personas y no será descargada en los repositorios de aplicaciones.

Otro punto importante que recalcaron fue que una aplicación debe ser específica, es decir que no abarque muchas funcionalidades. Esto porque puede ocurrir que una aplicación tenga veinticinco funciones, de las cuales talvez el usuario solo necesite tres, pero tenga que navegar diferentes menús para acceder a esas tres. Al hacer esto, el usuario probablemente va a perder bastante tiempo y se pierde la principal ventaja de los dispositivos móviles: acceder a la tecnología desde el bolsillo en cualquier momento y en poco tiempo.

En cuanto al desarrollo, algunos de los expertos comentaron que utilizaban los IDE's de cada plataforma, por lo que hacían las aplicaciones nativas. Mientras que otros prefieren utilizar librerías externas que permiten mayor portabilidad de las aplicaciones. Al hacer esto, probablemente se pueda sacrificar un poco de rendimiento ya que sería agregar una capa más, pero se ahorran tiempo de tener que rediseñar la aplicación para otra plataforma y abarcan un mayor mercado con el mismo producto.

Otras recomendaciones de los expertos, es que a la hora de empezar se hagan versiones con la menor cantidad de funciones posibles, es decir, ir poco a poco sin abarcar todos los requerimientos desde el principio del desarrollo. También recomendaron que para encontrar ideas de aplicaciones nuevas se pueden realizar competencias entre los compañeros fomentando la competencia sana.

## IV. BIBLIOGRAFÍA


[1] IBM (2006). RUP for Small Projects RPU, 1. Recuperado el 21 de Abril del 2013, de http://cgrw01.cgr.go.cr/rup/RUP.es/SmallProjects/index.htm#core.base_rup/guidances/supportingmaterials/welcome_2BC5187F.html.

[2] Krebs, J. (2007). The value of RUP certification DeveloperWorks, 1. Recuperado el 21 de Abril del 2013, de http://www.ibm.com/developerworks/rational/library/jan07/krebs/index.html.

[3] Barry W. Boehm, A Spiral Model of Software Development and Enhancement, Computer, May 1988, IEEE, pp.61-72

[4] Grady Booch, Ivar Jacobson, and James Rumbaugh, Unified Modeling Language 1.3, White paper, Rational Software Corp., 1998.

[5] R. E. Sorace, V. S. Reinhardt, and S. A. Vaughn, "High-speed digitalto-RF converter," U.S. Patent 5 668 842, Sept. 16, 1997.

[6] [1] Orjuela duarte, A. & Rojas c, M. (2008, 24 de Mayo). Las Metodologías de Desarrollo Ágil como oportunidad para la Ingeniería de Universidad de Pamplona Colombia, 1. Recuperado de http://www2.unalmed.edu.co/~pruebasminas/index.php?option=com_docman&task=doc_view&gid=340&tmpl=component&format=raw&Itemid=285

[7] [2] Kamthan, P. (2013). Extreme Programming for Mobile Aplicatons Concordia University, 277. Recuperado de http://www.irma-international.org/viewtitle/17089/

[8] [3] Maurer, F. & Martel, S. (2002). Extrene Programming Rapid Development for WEB AplicationsSpotlight, 1. Recuperado de http://cf.agilealliance.org/articles/system/article/file/1026/file.pdf

[9] [4] Biada, T. (2012, 20 de Febrero). Desarrollo aplicaciones móviles4veces más efectivas que en el PC | Blog de Agencia marketing online y consultora informática de Einnova Recuperado dehttp://www.einnova.com/notas/servicio/desarrollo-aplicacionesmoviles-efectivas

[10] [5] Carbo, R., Dorr, J. & Trapp, M. (s. f.). Focusing Extreme Programming on Usability Recuperado de http://subs.emis.de/LNI/Proceedings/Proceedings51/GIProceedings.51-34.pdf

[11] GeoSpatial (s. f.). Tipos de aplicaciones móviles Recuperado de http://geospatialtrainginges.com/recursos-gratuitos/tipos-deaplicaciones-moviles/

[12] Mobile Application Development Methodology V3 recuperado de: http://developer.smartface.biz/documents/Application_Development_Methodology.pdf el 8 de marzo del 2013.

[13] Cordero, J. (2011). METODOLOGIAS AGILES PROCESO UNIFICADO AGIL (AUP)UNIVERSIDAD UNION BOLIVARIANA CARRERA DE INGENIERIA DE SISTEMAS, 1. Recuperado de http://ingenieriadesoftware.mex.tl/images/18149/METODOLOGIAS%20AGILES.pdf.

[14] METODOLOGÍA DESARROLLO RÁPIDO DE APLICACIONES (S. F.). RECUPERADO EL 09 DE MAYO DEL 2013, DE http://mena.com.mx/gonzalo/maestria/ingsoft/presenta/rad/

[15] Schwaber, K. & Sutherland, J. (2013). *La guía de Scrum* Recuperado el 25 de Junio del 2013, de http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide%202011%20-%20ES.pdf#zoom=100

[16] Koch, A. (2004). *Agile Software Development : Evaluating the Methods for Your Organization* Norwood, MA, USA : Artech House .

[17] Holcombe, M. (2008). *Running an Agile Software Development Project* Hoboken, NJ, USA : Wiley .

[18] Pratap K.J Mohapatra, Mohapatra, P.K.J. "Software Engineering",New Age International Pvt Ltd Publishers (1 Dec 2009

[19] Sommerville, I. “Ingeniería del Software”. Séptima Edición. Pearson Education .S.A. Madrid, 2005.

[20] Amaro, S; Valverde, J. “Metodologías Ágiles”. Universidad Nacional de Trujillo. Perú, 2007.

TEC | Tecnológico de Costa Rica

GoTouch.

2012

# Centro de Investigaciones en Computación
# Tecnológico de Costa Rica

## Idea Inicial de App (IIA)

## Desarrollado por: xxxxxxx


Cartago, Costa Rica

# Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | xx/xx/20xx | xxxxxx | xxxxx |
| | | | |

# 1. Nombre de la app.

Nombre de la app

# 2. Creador(es).

Nombre de los creadores

# 3. Plataforma inicial.

iOS, Android, Windows Phone, Blackberry, otra.

# 4. Género.

Juego, app educativa, otro.

# 5. Dirigido a.

Público meta, edad, perfil de usuario.

# 6. Estilo visual.

2D, 3D.

# 7. Concepto.

Descripción general de la app.

En caso de un juego definir conceptos de jugabilidad.

# 8. Características.

Táctica, memoria, entretenimiento.

Posibilidades de ampliación, por ejemplo niveles.

Sistema de puntuación.

# 9. Aplicaciones similares.

Nombre y referencias de apps similares.

# 10. Vistas generales.

En caso de tenerse imágenes generales de la app.

# 11. Referencias.

Referencias en caso de haber sido utilizadas.

2012

**Centro de Investigaciones en Computación**
**Instituto Tecnológico de Costa Rica**

# Acta Constitutiva de Proyecto (ACP)

**Nombre del Proyecto**


Cartago, Costa Rica

# Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | 02/08/2012 | Samanta, Fabián, Sebastián, Mauricio, Andrés, Ariela | |
| | | | |

# Acta de constitución del proyecto

| | **Fecha:** xx/xx/xxxx | **Versión:** xx |
|---|---|---|
| | | |
| **Preparado por:** | xxxxxxxxxxxxxxxxxxxxxxx | |
| **Revisado por:** | xxxxxxxxxxxxxxxxxxxxxxx | |
| **Sponsor** | Centro de Investigaciones en Computación (CIC) | |
| **Cliente/Usuario Final** | ITCR – CIC | |
| **Contacto del Cliente** | xxxxxxxxxxxxxxxxxxxxxxx | |
| **Tipo de Proyecto:** | Productivo – Informático | |
| **Desarrollador líder:** | xxxxxxxxxxxxxxxxxxxxx | |
| **Unidades de Negocio:** | Clientes con dispositivos móviles compatibles con los productos generados. | |
| **Firma del responsable:** | | |

# Visión general del proyecto

## Descripción del proyecto

El propósito principal del proyecto consistirá en

## Justificación del Proyecto

Lo que motiva este proyecto es

## Metas y objetivos

El Proyecto tendrá como meta

Los objetivos del proyecto son los siguientes:

- Desarrollar
-

# Alcance del proyecto

| **El proyecto incluye.** |
|---|
| Desarrollo de |

| **El proyecto no incluye.** |
|---|
| Desarrollo |

| **Entregable** | **Criterio/s de aceptación** |
|---|---|
| xxxxxxxxxxxx. | Cumplir con |

| **Dependencias y Links con otros proyectos** |
|---|
| xxxxxxxxxx |

## Factores críticos de éxito

Presupuesto asignado

## Supuestos

Equipo

# Requerimientos de personal

| Puesto/Rol | Principal Interés en el Proyecto | Otras Responsabilidades |
|---|---|---|
| Administrador de iniciativa | Solicitar al desarrollador líder avances y/o informes del proyecto | -- |
| Desarrollador líder | El proyecto sea terminado con las expectativas y necesidades requeridas; bajo el presupuesto asignado. | Guiar el trabajo del equipo desarrollador, formular ideas y propuestas, facilitar la obtención de patrocinio de empresas terceras; y representar al equipo desarrollador ante la VIE, CIC y TEC. |
| Programador | | |
| Diseñador gráfico | | |

# Instalaciones y Recursos del Proyecto

| Recursos requeridos | Responsable |
|---|---|
| Ambiente de desarrollo especializado con xxxxxxxxxxxxxxx | xxxxxxxx: Aprobación de presupuesto. |
| Lugar físico de trabajo: xx Puestos de trabajo con cercanía física. | xxxxxxxx: Acondicionamiento del lugar de trabajo. |
| Repositorio centralizado de código fuente. | Equipo de Trabajo de desarrollo: Instalación y configuración del repositorio. |

# Hitos principales del proyecto

| Hito | Fecha |
|---|---|
| Entrega del Acta de Constitución de Proyecto. | xx/xx/xxxx |

TEC | Tecnológico de Costa Rica

GoTouch.

2012

Centro de Investigaciones en Computación
Instituto Tecnológico de Costa Rica

# Documento de Diseño de Aplicación (DDA)


Cartago, Costa Rica

# Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | xx/xx/xxxx | Xxxxxxxxxxxxxx | |
| | | | |

# 1. Introducción

## 1.1. Antecedentes del proyecto

En este punto se deben indicar los antecedentes del proyecto.

## 1.2. Alcance del proyecto

Debe especificarse de manera general el alcance del proyecto.

# 2. Información general

## 2.1. Nombre de la app

Se indica el nombre de la app.

## 2.2. Género de la app

En este punto se deben indicar el género de la app, si es una app educativa, un juego en cuyo caso debe clasificarse de acuerdo a la siguientes categorías:

- Arcade style
- Card, logic and board games
- Interactive Fiction
- Graphical Adventures
- Simulation
- Strategy
- First Person Shooter (FPS)
- Side Scrollers
- Third Person Shooters
- Role Playing Game (RPG)

## 2.3. Concepto

Se debe describir la idea general de la app y sus características en forma general.

## 2.4. Pilares

En este punto se deben indicar aspectos que caracterizan la app, tales como:

- Dificultad.
- Personalización del usuario.
- Opciones configurables.

## 2.5. Propósito y público objetivo

Se debe indicar el propósito del usuario y quiénes son los potenciales usuarios a los cuales va dirigido. En el caso de los usuarios potenciales es importante indicar el rango de edades.

## 2.6. Estilo visual

En este punto se debe describir el estilo visual que va a tener la app.

## 2.7. Preguntas básicas

### 2.7.1. Qué es la app?

Descripción de la app.

### 2.7.2. Donde está ambientada?

En este punto se describen los escenarios de la app.

### 2.7.3. Qué se controla en la app?

En este punto se indica que aspectos va a controlar el usuario en la app.

#### 2.7.4. Hay personajes? Cuáles son esos personajes?

Se definen de manera general los personajes, en caso que existan, que van a manejarse en la app.

#### 2.7.5. Cuál es el objetivo principal de la app?

En este punto se define el objetivo de la app para efectos del usuario.

### 2.8. Alcance.

En este punto se detallan los alcances de la app.

# 3. Detalle de la aplicación.

## 3.1. Jugabilidad / cambios en la app

En este punto si aplica se detalla la jugabilidad de la app.

## 3.2. Historia (storyboard)

La historia en la cual se basa la app.

## 3.3. Aspectos pedagógicos.

En este punto se detallan los aspectos pedagógicos que se pretenden abarcar en la app.

## 3.4. Flujo de la app.

Se debe explicar y al menos indicar en un diagrama el flujo que seguirá el usuario cuando utilice la app.

### 3.5. Movimientos e interacción

En este punto se detallan los movimientos y la interacción que va a tener el usuario con la app.

### 3.6. Voces / música / sonidos

En este punto se detallan todos los aspectos adicionales de la interfaz con el usuario.

# 4. Interfaz gráfica.

En este punto se detallan las principales pantallas que van a formar parte de la app.

### 4.1. Menú principal

### 4.2. Ayuda

### 4.3. Puntuaciones

### 4.4. Información sobre GoTouch

### 4.5. Pantallas propias de la app

# 5. Arquitectura de la app.

En este punto se describe la arquitectura bajo la cual se va a desarrollar la app.

# 6. Diseño de la app.

En este punto se indican todos los aspectos relacionados con el diseño de la app.

# 7. Referencias.

En este punto se indican referencias utilizadas.

# 8. Anexos.

## 8.1. Artes

En este punto se anexan los artes y diseños de la interfaz, en caso que existan.

# GUÍA PARA EL DESARROLLO DE UNA ARQUITECTURA PARA APLICACIONES WEB

Realizado por: Jeff Schmidt Peralta. Colaboradores: Esteban Carmona | Daniel Méndes | Ricardo Morales | Julio Agüero

Centro de Investigaciones en Computación, ITCR.

gotouch

# Tabla de contenido

# Propósito de este documento

Se realiza este documento con la intención de plasmar una **guía** para la creación de la arquitectura de aplicaciones M-Learning, además de brindar una serie de consideraciones importantes para el desarrollo de estas apps. Es realizado para el proyecto de investigación Arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles y pretende ser un apoyo para que los nuevos desarrollos tengan una guía que les indique qué se debe hacer primero, las opciones de componentes que tiene a disposición para realizar el tipo de aplicación deseada, consideraciones de diseño y de arquitectura que ayudarán a la toma de decisiones dentro del equipo y de esta forma se puede obtener un desarrollo más fluido, menos problemático y expedito.

Se determinaron las necesidades y la situación actual de un conjunto de proyectos en desarrollo en la iniciativa GoTouch y se pudo concluir la falta de un documento que logre brindar una base para empezar una aplicación más rápidamente y de manera organizada. Esta guía pretende satisfacer esas necesidades y apoyar a los desarrolladores a conseguir sus objetivos.

# Consideraciones generales de diseño para aplicaciones móviles

Las siguientes guías pretenden proveer información sobre aspectos que deben ser estudiados al diseñar una aplicación móvil. Debido a información recolectada por un grupo Quality Assurance, resulta relevante para el contexto del proyecto de investigación tomar en cuenta las consideraciones para lograr obtener un rendimiento y una eficiencia esperada de la aplicación.

- **Decidir si desarrollar un *rich client, thin web client* o una *rich internet application*.** Para estos efectos se debe saber que una aplicación móvil, por lo general, consta de varias capas que son: *user experience,* negocio y datos*. Rich client* significa que la capa de negocio y datos estará en el mismo dispositivo; mientras que *Thin client,* estas capas estarán en el servidor. En las aplicaciones desarrolladas dentro del marco del proyecto Arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles, la mayoría de las aplicaciones actualmente no necesitan de estar conectadas a Internet ni en comunicación con ningún servidor, por lo que se trata mayormente de aplicaciones *Rich Client*. Por último, si la aplicación requiere una interfaz de usuario compleja, acceso limitado a recursos locales y debe ser portable a otras plataformas, lo recomendable sería diseñar un *Rich Internet Application* client.

- **Determinar el tipo de dispositivos que se soportará**. Considerar tamaño de pantalla, resolución (DPI), las características de rendimiento del CPU, memoria y capacidad de almacenamiento.

- **Tomar en cuenta escenarios de poca conectividad y ancho de banda limitado**. Cuando la conectividad a redes es requerida, las aplicaciones deben manejar casos en los que la conexión a la red sea intermitente o no exista.

- **Diseñar una arquitectura en capas que sea apropiada para dispositivos móviles que promueva la reutilización y la mantenibilidad**. Utilizar el concepto de capas ayuda a maximizar la separación de responsabilidades, es decir, cada capa tiene una función específica y separada del resto. El diseño debe ser simplificado en comparación con una aplicación de escritorio o web, debido a las capacidades del dispositivo móvil

- **Considerar siempre limitaciones de recursos del dispositivo como duración de la batería, tamaño de la memoria y velocidad del procesador**. Todas las decisiones de diseño deben tomar en cuenta estos recursos. Usualmente la batería es uno de los factores más limitantes. *Backlighting* (luz de la pantalla), leer y escribir en memoria, conexiones inalámbricas, hardware especializado y velocidad del procesador tienen un impacto en el uso general del poder.

# 1. Áreas específicas de consideración

Existen varios asuntos comunes que se deben considerar cuando se está desarrollando el diseño de la aplicación. Se categorizarán en área específicas del diseño que son:

- Comunicación
- Gestión de la configuración
- Acceso a datos
- Manejo de excepciones
- Gestión del poder
- Pruebas
- Dispositivo
- Validación

## 1.1. Comunicación

La comunicación incluye comunicación inalámbrica (*over the air*) y comunicación alámbrica con un host PC, al igual que comunicación más especializada como Bluetooth o Infrarrojo (Infrared Data Association).

Se debe considerar las siguientes guías cuando se diseñe la estrategia de comunicación:

- Si se está desarrollando una aplicación que correrá en teléfonos móviles, considere cómo se debe comportar si se recibe una llamada durante la ejecución del programa. Diseñe la aplicación para que se suspenda y luego se reanude o, inclusive, se cierre la aplicación.
- Proteger la comunicación en conexiones no confiables como web services y otros métodos *over the air.*

## 1.2. Gestión de la configuración

Cuando se diseñe la gestión de la configuración de los dispositivos, se debe considerar cómo manejar *resets* del dispositivo.

Se debe considerar las siguientes guías cuando se diseñe la estrategia de gestión de la configuración:

- Diseñar para la restauración de la configuración después de un *reset* del dispositivo
- Por limitaciones de memoria, es preferible utilizar formato binario para archivos de configuración, en vez de XML.

## 1.3. Acceso a Datos

En el momento de diseñar el acceso a datos, considere cómo el bajo ancho de banda, la alta latencia y la intermitencia en la conectividad impactarán su diseño.

Considere las siguientes guías cuando se diseñe la estrategia de acceso a datos:

- Considerar la integridad de los datos. Los archivos que se dejan abiertos durante las suspensiones del dispositivo y las fallas de poder pueden causar problemas de integridad e datos. Incluir manejo de excepciones y lógica para asegurarse que las operaciones de archivos sean exitosas.
- Si se utiliza XML para almacenar o transferir datos, considere su tamaño total e impacto en el rendimiento. XML incrementa los requerimientos de ancho de banda y almacenamiento local. Utilice algoritmos de compresión o un método que no sea XML para transferencias.

## 1.4. Manejo de excepciones

Es de importancia para la seguridad y confiabilidad de la aplicación diseñar una estrategia efectiva para manejar las excepciones. Un buen manejo de excepciones previene que se muestren mensajes con información sensible al usuario, brinda robustez a la aplicación y ayuda a evitar un estado inconsistente en caso de un error.

Considere las siguientes guías cuando se diseñe la estrategia de acceso a datos:

- Diseñe para que la aplicación se recupere a un estado bueno luego de que ocurra una excepción.
- Atrape excepciones solamente si puede manejarlas y no utilice excepciones para controlar el flujo lógico.
- Diseñe un manejador (*handler*) para atrapar excepciones no manejadas.

## 1.5. Gestión del poder

Como se ha mencionado anteriormente, todas las decisiones de diseño deben tomar en consideración cuánto poder consume el dispositivo y su efecto en la duración de la batería.

Considere las siguientes guías cuando se diseñe la estrategia consumo de poder:

- Para conservar la vida de la batería, no actualice la interfaz de usuario cuando la aplicación está en *background* (el usuario no la está viendo).
- Para escoger protocolos de comunicación, se debe investigar sobre su consumo de poder.
- Considere apagar partes del dispositivo cuando no estén en uso o cuando no son requeridos. Ejemplos son la luz de la pantalla, discos duros, funciones de GPS, altavoces y comunicaciones inalámbricas.
- Cuando se use el protocolo de comunicaciones de hardware 3G, asegúrese de comunicarse en ráfagas (enviar un conjunto grande de datos por ocasión) y apagar la comunicación cuando no sea requerida.

## 1.6. Pruebas

El *debuggear* (depurar) las aplicaciones móviles puede ser mucho más costoso que hacerlo con una aplicación similar en una PC. Se debe considerar este costo cuando se decida cuáles y cuántos dispositivos soportará la aplicación.

Considere las siguientes guías cuando se diseñe la estrategia de depuración:

- Si se tiene un dispositivo que soporta la aplicación depure su código en el dispositivo en vez de un emulador.
- Si el dispositivo no está disponible, utilice un emulador para pruebas iniciales y depurar. Debe saber que un emulador podría ejecutar el código más lento que un dispositivo de verdad. Tan pronto como obtenga el dispositivo, corra el código en él.
- Pruebe escenarios en los que el dispositivo esté completamente desconectado de cualquier red o conexión, incluido estar desconectado de una sesión de depuración en una PC.

## 1.7. Específico del dispositivo

Se podría estar apuntando a varios tipos de dispositivo para una sola aplicación. Se debe tener presente la heterogeneidad en los distintos dispositivos a la hora de diseñar la aplicación. Factores a tomar en cuenta como tamaño de la pantalla y orientación, limitaciones en memoria y espacio de almacenamiento, ancho de banda y conectividad.

Considere las siguientes guías cuando se diseñe la estrategia de dispositivo:

- Crear código modular para permitir una sencilla exclusión de módulo de los ejecutables. Esto, para cubrir casos en los que archivos de ejecución más pequeños y separados sean requeridos debido a restricciones de memoria del dispositivo.
- Considere la inicialización perezosa (*lazy initialization*) para que los objetos sean sólo instanciados cuando sean requeridos.

## 1.8. Validación

Utilice validación para proteger al dispositivo y a la aplicación, y para mejorar la usabilidad.

Considere las siguientes guías cuando se diseñe la estrategia de validación:

- Validar los datos de entrada antes de enviarlos a un servidor puede reducir el *overhead* y mejorar el rendimiento. Esto también hace que la aplicación sea más responsiva cuando el usuario ingrese datos inválidos.
- Asegúrese de proteger los recursos de hardware como la cámara y la iniciación de llamadas de teléfono por medio de la validación de código y acciones que inicien estas características automáticamente.

# Capa de Presentación

Contiene los componentes encargados de implementar y desplegar la interfaz de usuario. Sumado a esto también se encarga de manejar la interacción que tenga el usuario con la interfaz. Esta capa a su vez es la encargada de controlar la entrada de datos por parte de los usuarios.

## 1. Componentes de la Capa de Presentación

### 1.1. Componentes de interfaz con el usuario (UI)

Son los componentes que le permiten al usuario interactuar con la aplicación. Además de ser los encargados de *renderizar* y darle formato a los datos que son mostrados a los usuarios de la aplicación. A su vez se deben encargar de validar los datos que son introducidos por los usuarios.

### 1.2. Componentes de proceso

Son los encargados de sincronizar, orquestar y organizar las interacciones del usuario. En muchos casos separar estos componentes resulta beneficioso, cuando se tiene una UI muy compleja.

## 2. Enfoque

Descripción de los pasos a seguir cuando se va a diseñar la capa de presentación. Estos pasos garantizan que se vayan a considerar todos los aspectos relevantes para el desarrollo de la arquitectura.

- *Identificar el tipo de cliente:* elija al cliente que satisfaga los requerimientos del sistema y a las limitaciones de este.
- *Determinar el modo en que se van a mostrar los datos:* es elegir el formato en que se presentarán los datos al usuario. Debe ir relacionado con el punto anterior, dependiendo el tipo de usuario así debe ser el modo en el que se le presentaran los datos.
- *Determinar la estrategia de validación de los datos:* dado que el usuario interactúa con la aplicación ingresando datos, es importante validar estos datos para evitar que se presenten problemas con la aplicación ocasionados por incompatibilidad con los datos ingresados.
- *Determinar su estrategia lógica de negocio:* presentar desde esta capa aspectos de la capa de negocio que sean importantes de presentar desde la capa de presentación. Los aspectos que se van a mostrar en esta capa deben estar bien identificados y bien justificada su utilización en esta capa. Para evitar confusiones y mezcla entre capas.
- *Determinar la estrategia para comunicarse con otras capas:* en el caso de que la aplicación utilice otras capas se debe desarrollar el método que van a utilizar estas para comunicarse, esta es una decisión que se toma desde la capa de Presentación debido a que esta es la primera línea de comunicación entre las diferentes capas.

## 3. Consideraciones de Diseño

Factores críticos de éxito que se deben considerar cuando se diseña la capa de presentación.

- *Elegir la apropiada tecnología para elaborar la IU:* se debe definir, basado en las características del cliente, el tipo de tecnología que se va a usar, si se va a ser una capa rica en acceso a internet, un cliente web o un usuario inteligente.
- *Utilizar patrones:* revisar cuales patrones podrían servir como solución a uno o varios de los problemas que se podrían presentar (al final del documento hay una lista de los posibles patrones que se pueden utilizar)
- *Diseñar de manera separada los componentes de UI con los componentes de proceso:* utilizar componentes dedicados únicamente a UI, de esta manera se centrara la atención a aspectos exclusivos de la capa de presentación como la manera de presentar la información, la interacción, entre otras; y a su vez evitar que se acoplen los componentes encargados de presentar interfaz con los encargados de los procesos.
- *Diseño orientado al usuario:* un aspecto clave antes de realizar el diseño de esta capa es comprender al usuario. Para esto se pueden utilizar entrevistas, encuestas, estudios de usabilidad, entre otros; que determinen cuál es el mejor método de presentar la información.

# 4. Marco de la Capa de Presentación

A lo largo del tiempo se ha logrado encontrar una serie de errores en los que se incurre cuando se desarrolla esta capa. Estos errores se han categorizados, y a continuación se presenta una serie de pasos que se deben seguir o de aspectos a tomar en cuenta para evitar caer sobre estos errores característicos de la etapa.

## 4.1. Almacenamiento en Cache

El almacenamiento en cache es una de los mejores mecanismos para mejorar el desempeño de las aplicaciones. Pero estos datos almacenados en cache deben tener un objetivo bien definido y no almacenar todo por almacenarlo. Aspectos a considerar a la hora de diseñar la estrategia de almacenamiento en cache:

- Los recursos siempre son limitados, por lo que se debe siempre tomar en cuenta para elaborar la estrategia.
- No almacenar en cache datos volátiles.
- Almacenar datos que estén listos para usarse, y no datos que deben ser procesados, ejemplo: es mejor almacenar en cache un objeto instanciado en vez de información de base de datos.
- No almacenar en cache datos sensibles a menos que estén encriptados.
- Que la aplicación no dependa de la información almacenada en la cache, la cache se va a eliminar o incluso puede que ya se haya eliminado.

## 4.2. Administración de Excepciones

Utilizar un mecanismo centralizado que se encargue que capturar y responder a las excepciones de manera consistente. Se debe prestar especial atención a las excepciones que se propagan a través de las capas o de sus límites. Diseñar las excepciones para que no impacten a la aplicación ni expongan información sensible.

- Utilizar mensajes amigables para notificar al usuario que ha ocurrido un error.

- No se debe exponer datos importantes en las páginas de error, mensajes de error, archivos de registro o archivos de auditoría.
- Diseñar una excepción global que se encargue de mostrar los mensajes de error, las páginas de error y todas las excepciones.
- No utilizar excepciones para controlar la lógica de la aplicación.

### 4.3. Input

Diseñar el método de ingreso de datos, en los requerimientos de la aplicación. Para tener un mayor grado de usabilidad, se recomienda utilizar mejores prácticas establecidas por la industria.

- Utilizar formularios que controlen los accesos de datos, en las tareas de recolección de datos.
- Utilice un asistente, para el usuario, en las tareas de recolección de datos que son más complejas.
- Considerar la accesibilidad en el diseño, considerar personas con discapacidades.

### 4.4. Layout

Diseñar la interfaz de tal forma que los componentes sean independientes de los procesos de la UI. Para esto se busca diseñar una interfaz que no necesite código de procesos necesariamente, sino aspectos centrados únicamente en el despliegue de la interfaz. Considerar también el dispositivo, dado que las pantallas de los móviles son pequeñas, no se puede mostrar en una sola todas las funcionalidades, es por esto que se torna importante la utilización de patrones. Se debe buscar que la interfaz sea lo más consistente posible, así como la aplicación.

- Utilizar plantillas que le brinden a la aplicación una apariencia similar en todas sus pantallas, sin importar cuál sea la función o proceso que se está realizando dentro de la aplicación.
- Utilizar un diseño en común en toda la aplicación para facilitar la accesibilidad y usabilidad.
- Considerar los métodos que tiene el móvil para la interacción como touch screen.
- Utilizar patrones para separar la interfaz de usuario con los procesos de esta.

### 4.5. Navegar

Desarrollar la estrategia de navegación de tal forma que le permita al usuario navegar de manera fácil a través de las diferentes pantallas o páginas. Asegurarse que los controles y enlaces través de la aplicación son consistentes, para evitar la confusión del usuario.

- Utilizar patrones de diseño conocidos para separar la UI de la lógica de navegación, cuando la lógica es compleja. En móviles utilizar siempre una “home page” y a partir de ahí, en método de cascada, llegar a las demás partes de la aplicación, siempre volviendo a la pantalla principal.
- Una barra de tareas y un menú bien diseñado ayudan al usuario a encontrar con mayor facilidad las distintas funcionalidades de la UI.
- Hacer de la navegación lo más predecible que se pueda.
- Determinar como la aplicación va a preservar el estado de la navegación, en caso que la aplicación tenga que preservar el estado entre las sesiones.

### 4.6. Procesamiento de Solicitudes

Diseñar el procesamiento de solicitudes, el código de mantenimiento y las pruebas con la respuesta del cliente en mente.

- Utilizar operaciones asincrónicas o subprocesos de trabajo para evitar que se bloquee la interfaz cuando se realicen tareas de larga duración.

### 4.7. Experiencia del Usuario.

Tomar en cuenta estudios de usabilidad, entrevistas y encuestas que le permitan entender que quiere el usuario y que espera de la aplicación. Desarrolle la UI con los resultados de estos estudios en mente

- No desarrollar interfaces demasiado sobrecargadas o complejas. Siempre se debe proporcionar un camino claro a través de la aplicación.
- Se debe diseñar de tal manera que se permita o soporte la personalización por parte del usuario.
- Permitir que el usuario controle como desea interactuar con la aplicación, y la forma en que se le muestran los datos.

### 4.8. Componentes de UI

Son los controles y componentes utilizados para mostrar la información al usuario y aceptar los datos ingresados por este.

- Crear controles personalizados, o usar controles de terceros, para usar controles personalizados solamente si se tienen tareas de recolección de datos.
- Buscar ampliar las funciones de los controles existentes en vez de buscar crear controles nuevos.

### 4.9. Componentes de procesamiento de UI

Sincronizan las interacciones del usuario. No son siempre necesarias, se utilizan solo en el caso en que se necesita un rendimiento significante a la hora de procesar la capa de presentación. Se debe tener el cuidado de no mezclar la lógica del negocio con la lógica de presentación

- No crear componentes de procesamiento a menos que sean 100% necesarios
- En caso de que la UI necesite de procesamiento complejo o requiera comunicarse con otras capas, usar estos componentes para desacoplar este procesamiento de la interfaz de usuario.
- Dividir los componentes de procesos en tres roles distintos: modelo, vista, controlador
- Utilizar patrones.

### 4.10. Validación

Diseñar una efectiva validación de los datos ingresados es una estrategia crítica para la seguridad de la aplicación. Genera las reglar necesarias para este proceso de validación, y que se vayan a utilizar en futuras aplicaciones.

- Validar todos los datos ingresados para reducir errores.

- Diseñar la estrategia de validación para restringir, rechazar y limpiar las entradas maliciosas de datos.

## 4.11. Guía de Patrones

Los patrones están divididos en las mismas categorías ya reseñadas. Se recomienda considerar estos patrones cuando se vaya a diseñar la capa de presentación:

- Llamada asincrónica (Asynchronous Callback): ejecuta tareas largas (long-running task) en hilos separados que se ejecutan en segundo plano, y provee la función de llamar al hilo cuando la tarea ha sido terminada.
- Dependencia de la Cache (Cache Dependency): utiliza información externa para determinar el estado de los datos almacenados en el cache.
- Cadena de responsabilidades (Chain of Responsibility): evita el acoplamiento de una solicitud con el elemento encargado de responder, para esto utiliza más de un objeto para gestionar las diferentes solicitudes.
- Patrón de comando (Command Pattern): encapsula el proceso de solicitud en objetos de comandos independientes con interfaz de ejecución en común.
- Entidad de traducción (Entity Translator): un objeto transforma el tipo de los datos ingresados, a los tipos que los procesos de negocio pide; y realiza el proceso inverso para realizar la respuesta.
- Escudo de excepción (Exception Shielding): provee un servicio que muestra la información de cómo se implementa las excepciones, internamente, cuando ocurren.
- MVC (Model-View-Controller): separa el código de la UI en tres unidades: modelo (datos) vista (interfaz) controlador (procesos lógicos), enfocado en la vista.
- Control de página (Page Controller): acepta una solicitud de entrada y la maneja para una específica página o acción en un sitio web.
- Modelo de Presentación (Presentation Model): mueve todas las vistas lógicas y estados fuera de la vista, y traduce la vista a través de datos ocultos y plantillas.
- Plantillas de vista (Template View): implementa plantillas similares de vista y construye nuevos puntos de vista usando estas plantillas como base.

## 4.12. Mapeo de los Patrones con las categorías

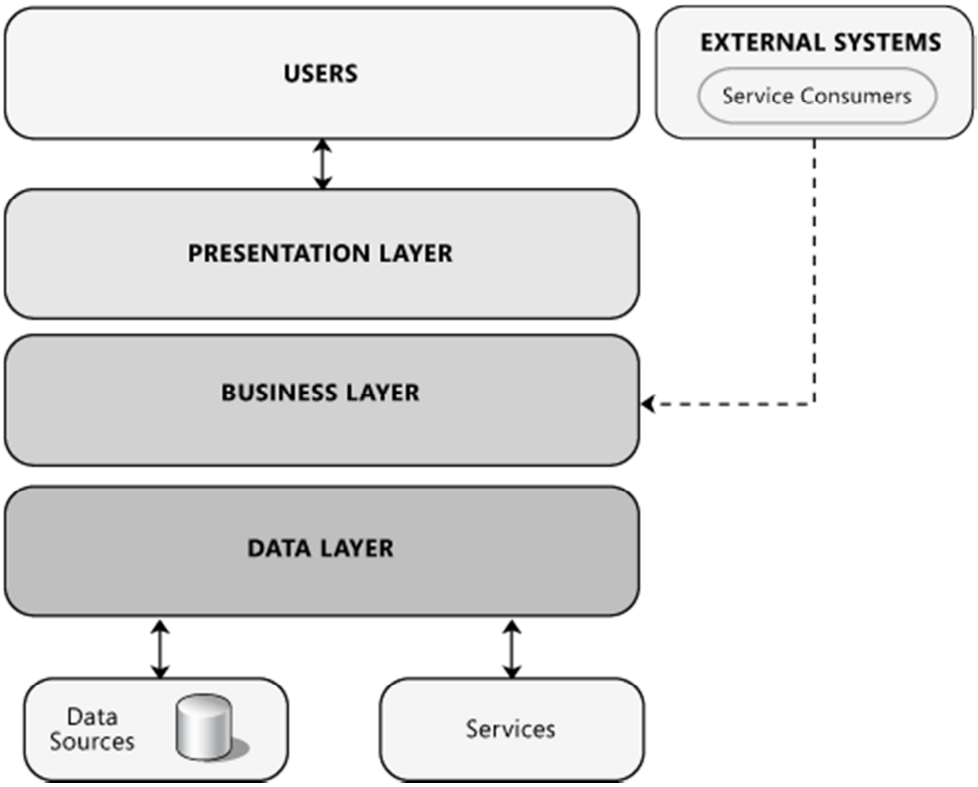
En la siguiente tabla se encuentra una relación que muestra cuáles patrones, de los presentados anteriormente, pueden ser usados en las categorías explicadas a lo largo del documento.

| Categoría | Patrón relacionado |
|---|---|
| **Almacenamiento en Caché** | Cache Dependency |
| **Administración de Excepciones** | Exception Shielding |
| **Layout** | Template View |
| **Navegar** | Command Pattern |
| **Presentación** | Entity Translator |
| **Experiencia del Usuario** | Asynchronous Callback<br>Chain of Responsability |
| **Componentes de** | Model-View-Controller (MVC) |

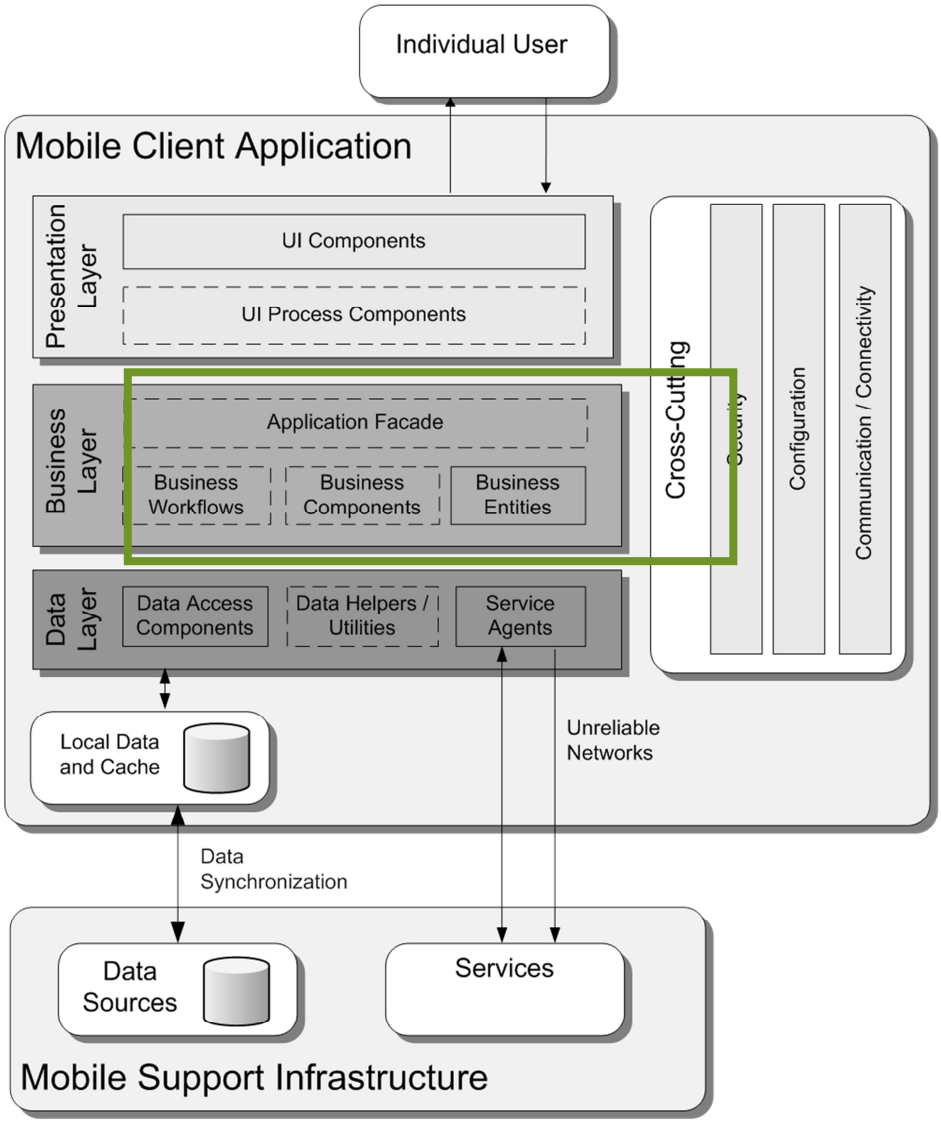
| | |
|---|---|
| **procesamiento de UI** | Presentation Model |

# Capa de Negocio

Esta capa implementa la funcionalidad básica del sistema, y encapsula la lógica de negocio relevante. Por lo general, se compone de varios componentes, algunos de los cuales pueden exponer interfaces de servicio que otras pueden llamar. Para ciertas aplicaciones que requieren acceso a datos y servicios externos remotos, estos son consumidos por la capa de negocio a través de servicios Web.



La siguiente imagen representa como la capa de negocio interactúa con las demás capas, así como los componentes mínimos que debe tener.




## 1. Componentes clave del negocio

A continuación se describen los roles y responsabilidades de los principales componentes dentro de la capa de negocio.

1. **Fachada de la aplicación (opcional):** combina múltiples operaciones de negocio en una sola operación basada en mensajes. Es posible acceder a la fachada de la aplicación a través de la capa de presentación mediante el uso de diferentes tecnologías de comunicación.
2. **Componentes de negocio:** Dentro de la capa de negocio hay diferentes componentes que proporcionan servicios de negocio, tales como el procesamiento de reglas de negocio e formas de interactuar con los componentes de acceso a datos.

3. **Las entidades de negocio:** Los componentes de negocio que se utilizan para pasar los datos entre los demás componentes de negocio son considerados entidades de negocio. Los datos se pueden representar entidades de negocio del mundo real, tales como productos y pedidos, o las entidades de base de datos, como tablas y vistas. Las entidades empresariales se pueden implementar utilizando estructuras de datos tales como bases de datos y lenguaje de marcado extensible (XML).
4. **Flujos de trabajo de negocio:** Muchos procesos de negocio implican varios pasos que se deben realizar en el orden correcto y orquestado. Flujos de trabajo empresariales definen y coordinan los procesos empresariales de larga ejecución, de múltiples pasos, y pueden ser implementados utilizando herramientas de gestión de procesos de negocio.

Con el fin de minimizar la huella en el dispositivo, las aplicaciones móviles generalmente usan enfoques de estratificación menos rígida y un menor número de componentes discretos.

## 2. Enfoque a la capa de negocio

Con el fin de poder diseñar los componentes de la capa de negocio se deben de llevar a cabo las siguientes actividades clave, estas actividades deben ser llevadas a cabo simultáneamente con el diseño de la capa de datos.

### 2.1. Crear un diseño global de su capa de negocio:

- Identificar a los consumidores, es decir a los usuarios a los que se va a enfocar la aplicación y como ellos van a utilizarla, de la capa de negocio. Ya sean niños, personas adultas mayores, público en general por ejemplo.
- Determinar cómo va a exponer la capa de negocio.
- Determinar los requisitos de seguridad de la capa de negocio.
- Determinar las necesidades y la estrategia de validación para la capa de negocio.
- Determinar la estrategia de manejo de excepciones para la capa de negocio.

### 2.2. Diseñar los componentes de negocio:

- Identificar los componentes de negocio que la aplicación va a utilizar.
- Asegurarse de tomar decisiones clave sobre la ubicación, acoplamiento, y las interacciones de los componentes de negocio.
- Elija el soporte de transacciones apropiado. Es decir cómo se va a realizar el envío de mensajes entre los componentes de la aplicación.
- Identificar cómo se manejan las reglas de negocio.
- Identificar los patrones que se ajustan a los requisitos.

### 2.3. Diseñar componentes de entidad de negocio:

- Identificar los formatos de datos comunes para las entidades de negocio.
- Elija el formato de datos.
- Opcionalmente, elija un diseño para los objetos personalizados.

### 2.4. Diseñe los componentes de flujo de trabajo:

- Identificar el flujo de trabajo utilizando escenarios.

- Elija un modo de edición. Esto debido a que las aplicaciones deben de poder ser actualizadas y mejoradas conforme pase el tiempo. Con un modo de edición estandarizado se pueden realizar cambios sin que otros componentes se vean afectados.
- Determinar cómo se responderá a las normas.
- Elija una solución de flujo de trabajo.
- Diseñe los componentes de negocio para apoyar el flujo de trabajo.

## 3. Consideraciones de diseño

En el diseño de una capa de negocio, el objetivo de un arquitecto de software es minimizar la complejidad mediante la separación de las tareas en diferentes áreas de interés. Por ejemplo, las entidades de procesamiento de negocios, flujo de trabajo y las entidades de negocios todos representan diferentes áreas de interés. Por las ventajas del bajo acoplamiento, como reutilización t buenas prácticas de diseño se deben tomar las siguientes pautas en el diseño de la capa de negocio:

- **Decidir si se necesita una capa de negocios separada**. Es una buena idea usar una capa de negocios separada, siempre que sea posible para mejorar la capacidad de mantenimiento de la aplicación.
- **Identificar las responsabilidades de la capa de negocio**. Use una capa de negocio para el procesamiento de reglas de negocio complejas, transformación de datos, la aplicación de las políticas, y para su validación.
- **No mezcle diferentes tipos de componentes en la capa de negocio.** Usar una capa de lógica de negocio para desacoplarla de la presentación y el código de acceso a datos, esto para simplificar las prueba de la lógica de negocio.
- **Reutilizar la lógica de negocio común.**
- **Evite estrecho acoplamiento entre las capas**.

## 4. Componentes de Negocio

Los componentes de negocio implementan las reglas de negocio en diversos patrones y aceptan y devuelven estructuras de datos simples o complejos. Los componentes de negocio deben exponer la funcionalidad de una manera que sea transparente a los conjuntos y servicios de datos necesarios para realizar el trabajo.

Es importante tener en cuenta las siguientes pautas en el diseño de componentes de negocio:

- Evite mezclar la lógica de acceso a datos y la lógica de negocio dentro de los componentes de negocio.
- Diseñe componentes para ser altamente cohesivos y bajo acoplamiento.
- Si se desea mantener las reglas de negocio independientes de los datos de negocio, se debe considerar el uso de componentes de procesos de negocio para implementar las reglas de negocio.
- Si la aplicación tiene reglas de negocio volátiles, es necesario guardarlas en un motor de reglas.

## 5. Concurrencia y transacciones

En el diseño de la concurrencia y transacciones, por tanto es importante identificar el modelo de concurrencia apropiado y determinar cómo se va a gestionar las transacciones. Se puede elegir entre un modelo optimista y un modelo de concurrencia pesimista. Con la concurrencia optimista, los bloqueos no se llevan a cabo en los datos y actualizaciones requieren un código de verificación, por lo general contra una marca de tiempo, para verificar que los datos no han cambiado desde que se recibieron por última vez. Con simultaneidad pesimista, los datos se bloquean y no se pueden actualizar por otra operación hasta que se libere el bloqueo.

## 6. Acceso a datos

Diseñar una estrategia de acceso a datos eficaz para la capa de negocio es importante para maximizar la facilidad de mantenimiento y la separación de preocupaciones. Si no se hace puede dificultar la gestión y la ampliación si los requisitos del negocio cambian.

Tener en cuenta las siguientes pautas en el diseño de una estrategia de acceso a datos:

- Evitar mezclar el código de acceso a datos y la lógica de negocio dentro de los componentes de negocio.
- Evitar acceder directamente a la base de datos de la capa de negocio.
- Considere el uso de una capa de acceso de datos independiente para el acceso a la base de datos. Según lo redactado en la sección: ***Capa de Acceso a Datos.***

## 7. Registro e instrumentalización (Logging)

Diseñar un buen registro para la capa de negocio es importante para la seguridad y la fiabilidad de la aplicación. Si no se hace puede la aplicación vulnerable a las amenazas.

Tener en cuenta las siguientes pautas en el diseño de una estrategia de instrumentación y registro:

- Centralizar el registro y la instrumentación de la capa de negocio.
- No conservar información sensible en los archivos de registro.
- Asegurarse de que la falta de registro no afecta al funcionamiento normal de la capa de negocio.
- Considere la auditoría y el registro de todos los accesos a las funciones dentro de la capa de negocio.

## 8. Validación

El diseño de una solución eficaz para la validación de la capa de negocio es importante para la seguridad y la fiabilidad de la aplicación. De lo contrario, puede dejar la aplicación vulnerable a ataques de cross-site scripting, ataques de inyección de SQL, desbordamientos de buffer, y otros tipos de ataques de entrada.

Tener en cuenta las siguientes pautas en el diseño de una estrategia de validación:

- Validar todos los parámetros de entrada dentro de la capa de negocio, incluso cuando la validación de entrada se produce en la capa de presentación.
- Unifique se método de validación, si es que se puede reutilizar.
- Validar los datos de entrada en cuanto a longitud, distancia, tamaño y tipo.

## 9. Flujos de Trabajo

Los componentes de flujo de trabajo sólo se utilizan cuando la aplicación debe ser compatible con una serie de tareas que dependen de la información que se procesa. En el diseño de los componentes de flujo de trabajo, es importante tener en cuenta cómo va a gestionar los flujos de trabajo, y para comprender las opciones disponibles.

Tener en cuenta las siguientes pautas en el diseño de una estrategia de flujo de trabajo:

- Implementar flujos de trabajo dentro de los componentes que implican un proceso de múltiples pasos o de larga duración.
- Seleccione un estilo de flujo de trabajo adecuado en función del escenario de aplicación.
- Manejar condiciones de falla dentro de los flujos de trabajo, y exponer las excepciones adecuadas.
- Si el componente tiene que ejecutar un conjunto específico de pasos de forma secuencial y sincronizada, considere el uso del patrón pipeline
- Si los pasos del proceso se pueden ejecutar de forma asincrónica en cualquier orden, puede utilizar el patrón de evento (event pattern).

## 10.Acoplamiento

El acoplamiento entre clases o subsistemas es una medida de interconexión entre esas clases o subsistemas. El acoplamiento fuerte significa que las clases relacionadas necesitan saber detalles internos unas de otras, los cambios se propagan por el sistema y el sistema es más difícil de entender.

Por tanto el acoplamiento fuerte genera varios problemas:

- El primer problema es una cierta dificultad en entender el código debido al modo en que se mezclan las diferentes componentes.
- El segundo problema es que cualquier cambio en la estrategia de acceso a datos, la estructura de la base de datos o las estrategias de configuración se propagarán también por el código de la lógica de negocios. Es decir la lógica de negocios sabe demasiado acerca de la infraestructura subyacente.
- El tercer problema es que no podemos reutilizar el código de la lógica de negocios independientemente de la estructura de la base de datos por ejemplo. Tampoco es posible reutilizar la funcionalidad de acceso a datos incorporada. El acoplamiento entre el acceso a datos y la lógica de negocios puede que no sea un problema, pero ¿y si quisiéramos adaptar la lógica de negocios para su uso con datos introducidos por analistas directamente en una hoja de cálculo de Excel? ¿Y si quisiéramos probar o depurar la

propia lógica de negocios? No podemos hacer nada de eso dado que la lógica de negocios tiene un acoplamiento fuerte con el código de acceso a datos. Sería mucho más fácil modificar la lógica de negocios si pudiéramos aislarla de las otras cuestiones. Como se propone en la capa de acceso a datos.

En resumen, los objetivos detrás de la obtención de acoplamiento débil entre clases y módulos son:

- Hacer que el código sea más fácil de leer.
- Hacer que nuestras clases sean más fáciles de consumir por otros desarrolladores, ocultando la parte "fea" del funcionamiento interno de nuestras clases en API bien diseñadas.
- Aislar posibles cambios en un área pequeña del código.
- Reutilizar clases en contextos completamente nuevos.

## 11.Cohesión

La cohesión se define como una medida de proximidad en la relación entre todas las responsabilidades, los datos y los métodos de una clase. Es decir la medida que indica si una clase tiene una función bien definida dentro del sistema.

Una prueba fácil de cohesión consiste en examinar una clase y decidir si todo su contenido está directamente relacionado con el nombre de la clase y descrito por el mismo. Las clases con nombres vagos, no cuentan. Si la clase tiene responsabilidades sin relación con su nombre, probablemente pertenecerán a otra clase. Si encuentra un subconjunto de métodos y campos que se podría agrupar fácilmente aparte, con otro nombre de clase, tal vez deba extraer esos métodos y campos a una nueva clase. Es por tanto que hacer pruebas sobre si existe una alta cohesión es importante.

## 12.Mapeo de Patrones a utilizar

Patrones principales están organizados por las principales categorías detalladas en la tabla siguiente. Considere el uso de estos patrones para tomar decisiones de diseño para cada categoría.

| Categoría | Patrones Relevantes |
|---|---|
| **Componentes de Negocio** | Fachada de aplicación<br>Cadena de responsabilidad<br>Comando |
| **Entidades de negocio** | Entity Translator<br>Table Module |
| **Concurrencia y Transacciones** | Capture Transaction Details<br>Coarse-Grained Lock<br>Implicit Lock<br>Optimistic Offline Lock |

| | |
|---|---|
| | Pessimistic Offline Lock<br>Transaction Script |
| **Acceso a Datos** | Active Record<br>Query Object<br>Row Data Gateway<br>Table Data Gateway |
| **Flujos de Trabajo** | Flujo de trabajo basada en datos<br>Flujo de trabajo humano<br>Flujo de trabajo secuencial<br>State-driven workflow |

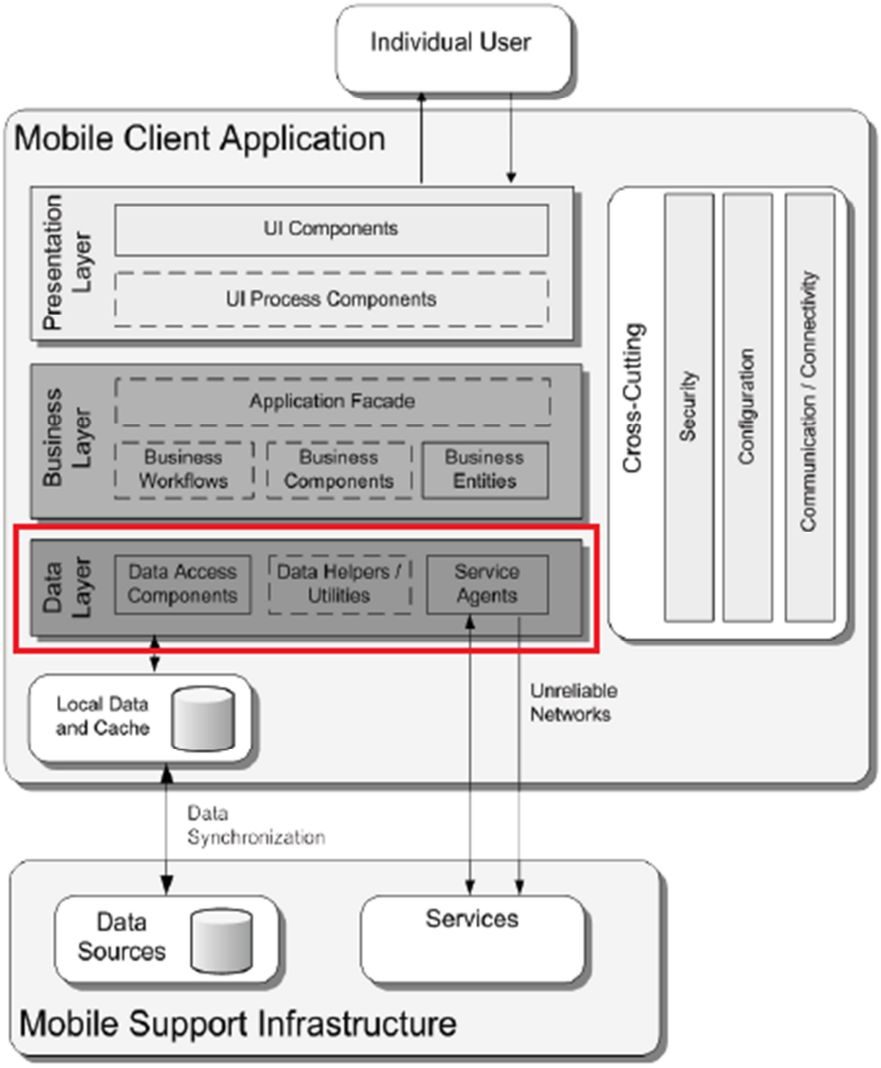
## 13.Descripción de los Patrones

- **Active Record**: Incluye un objeto de acceso a datos dentro de una entidad de dominio.
- **Capture Transaction Details**: Crea objetos de base de datos, tales como desencadenadores y tablas sombra, para registrar los cambios en todas las tablas que pertenecen a la transacción.
- **Chain of Responsibility**: Evita el acoplamiento del remitente de una solicitud a su receptor por lo que permite más de un objeto para manejar peticiones.
- **Coarse Grained Lock:** Bloquea un conjunto de objetos relacionados con un solo bloqueo.
- **Comando**: Encapsula el procesamiento de una solicitud en un objeto de comando separado con una común interfaz de ejecución.
- **Flujo de trabajo basada en datos:** Un flujo de trabajo que contiene las tareas cuya secuencia está determinada por los valores de los datos en el flujo de trabajo o el sistema
- **Flujo de trabajo humano:** Un flujo de trabajo que implica tareas realizadas manualmente por humanos.
- **Implicit Lock:** Utilice un marco de referencia para adquirir bloqueos en nombre de código que tiene acceso a los recursos compartidos.
- **Optimistic Offline Lock**: Asegura que los cambios realizados por una sola sesión no entren en conflicto con los cambios realizados por otro período de sesiones.
- **Pessimistic Offline Lock**: Evitar conflictos forzando una transacción a obtener bloqueo en los datosantes de usarlo.
- **Query Object:** Un objeto que representa una consulta de base de datos.
- **Row Data Gateway:** Un objeto que actúa como una puerta de entrada a un único registro de un origen de datos.
- **Flujo de trabajo secuencial:** Un flujo de trabajo que contiene las tareas que siguen una secuencia, donde se inicia una tarea después de la finalización de la tarea anterior.
- **State-driven Workflow:** Un flujo de trabajo que contiene las tareas cuya secuencia se determina por el estado del sistema.
- **Table Data Gateway**: Un objeto que actúa como una puerta de entrada a una tabla o vista en una fuente de datos y centralizar toda las consultas de selección, inserción, actualización y eliminado.
- **Table Module**: Un único componente que se encarga de la lógica de negocio para todas las filas de una tabla de base de datos o vista.

- **Transaction Script**: Organiza la lógica de negocio de cada transacción en un solo procedimiento, realiza llamadas directamente a la base de datos a través de “thin database wrapper”.

# Capa de Acceso a Datos

A continuación se describen las instrucciones clave para diseñar la capa de datos de una aplicación. La siguiente figura muestra como la capa de datos se ajusta a la arquitectura típica de una aplicación.




## 1. Componentes de la Capa de Datos

**Componentes lógicos del acceso a datos**: los componentes de acceso a datos abstraen la lógica necesaria para acceder a los almacenes de datos subyacentes. Centraliza la funcionalidad de acceso a datos, lo que hace que la aplicación sea más fácil de mantener y configurar.

**Componentes de ayuda o servicios:** la mayoría de las tareas de acceso a datos requieren cierta lógica común que puede ser extraída e implementada en un componente separado y reutilizable. Esto ayuda a simplificar la complejidad de los componentes de acceso a datos y sobre todo, minimiza el volumen de código a mantener.

Se componen de bibliotecas especializadas y rutinas personalizadas especialmente diseñadas para maximizar el rendimiento de acceso a datos y reducir el desarrollo de requerimientos de los componentes lógicos y las partes de agente de servicio de la capa.

**Agentes de servicio:** cuando un componente de negocio debe usar funcionalidad expuesta por un servicio externo (servicios brindados por un proveedor externo a la organización) como por ejemplo el servicio de correo de Google, se necesitara crear código que administre la semántica de comunicación con ese servicio. Los Agentes de Servicios aíslan dicha idiosincrasia de forma que, manteniendo ciertas interfaces, sería posible sustituir el servicio externo original por un segundo servicio diferente sin que nuestro sistema se vea afectado.

## 2. Guía para diseñar la capa de acceso a datos

Un correcto enfoque para diseñar la capa de datos reducirá el tiempo de desarrollo y el mantenimiento de la capa de datos después de que la aplicación es desarrollada. Esta sección describe brevemente un enfoque de diseño efectivo para la capa de datos. Desarrollar las siguientes actividades claves cuando diseñamos la capa de datos:

1. **Crear un diseño global de la capa de acceso a datos:**
    a. Identifique los requerimientos de las fuentes de datos.
    b. Determine el método de acceso a datos.
    c. Busque como mapear las estructuras de datos a las fuentes de datos.
    d. Determine como conectarse a las fuentes de datos.
    e. Determine las estrategias para el manejo de errores de las fuentes de datos.
2. **Diseñe los componentes de acceso a datos:**
    a. Enumerar las fuentes de datos que se van a acceder.
    b. Determinar el método de acceso para cada fuente de datos.
    c. Determinar si los componentes de ayuda son necesarios o deseables para simplificar el acceso a los datos.
    d. Determinar los patrones de diseño relevantes.
3. **Diseñe los componentes de ayuda o servicios:**
    a. Identifique la funcionalidad que puede ser removida fuera de los componentes de acceso a datos y centralícela para ser reutilizada.
    b. Busque librerías disponibles de componentes de ayuda o servicios.
    c. Identifique y cree componentes de ayuda para problemas comunes, tales como strings de conexión, autenticación de las fuentes de datos, monitoreo y excepciones.
    d. Considere implementar rutinas para el control de acceso a datos y pruebas en los componentes de ayuda.
4. **Diseñar los agentes de servicio:**
    a. Use la herramienta apropiada para añadir una referencia de servicio. Esto generará un proxy y las clases de datos que representan los contratos de datos desde el servicio.

b. Determine como el servicio será utilizado en la aplicación. Para la mayoría de las aplicaciones, se debe usar una capa de abstracción entre la capa de negocio y la capa de acceso a datos, la cual proporcionara una interfaz consistente independientemente de la fuente de datos. Para aplicaciones pequeñas, las capa de negocio o incluso la capa de presentación, puede acceder al agente de servicios directamente.

## 3. Instrucciones de diseño

Las siguientes instrucciones de diseño proveen información acerca de diferentes aspectos de la capa de acceso a datos que usted debería considerar. Seguir estas instrucciones para asegurar que su capa de acceso a datos cumple con los requerimientos de la aplicación, está desarrollada de manera eficiente y segura, y es fácil de mantener y ampliar cuando los requisitos del negocio cambian.

**Elija la tecnología de acceso a datos.** Elegir una apropiada tecnología de acceso a datos dependerá del tipo de datos que se esté analizando y como usted quiere manipular los datos dentro de la aplicación.

**Utilice una abstracción para implementar una interfaz de bajo acoplamiento para la capa de acceso a datos.** Esto puede ser logrado mediante la definición de componentes de interfaz, tales como una puerta de enlace con entradas y salidas conocidas, que traduce las peticiones a un formato comprensible por los componentes de la capa.

**Encapsular la funcionalidad de acceso a datos dentro de la capa de acceso a datos.** La capa de acceso a datos oculta los detalles de acceso a las fuentes de datos. Es responsable de la gestión de las conexiones, la generación de consultas y de mapear las entidades de la aplicación a las estructuras de las fuentes de datos. Los consumidores de la capa de acceso a datos interactúan a través de interfaces abstractas usando entidades de la aplicación tales como objetos personalizados y XML. Las otras capas de la aplicación manipulan estos datos de maneras más complejas para implementar la funcionalidad de la aplicación.

**Decidir cómo mapear las entidades de la aplicación a las estructuras fuentes de datos.** El tipo de entidad que utiliza en su aplicación es el principal factor en la decisión de como mapear estas entidades a las estructuras fuentes de datos.

**Decidir cómo se gestionaran las conexiones.** Como regla general, la capa de acceso a datos debe crear y gestionar todas las conexiones a todas las fuentes de datos requeridas por la aplicación. Se debe elegir un método apropiado para el almacenamiento y protección de la información de las conexiones que se ajuste a los requisitos de seguridad de las aplicaciones.

**Determine como se van a manejar las excepciones de datos.** La capa de acceso a datos debe de capturar y manejar todas las excepciones asociadas con las fuentes de datos y las operaciones CRUD (create, read, update, delete). Excepciones relativas a los datos, acceso a los datos y errores de tiempos de espera. Estas excepciones se pasan a otras capas si las fallas afectan la capacidad de respuesta de la aplicación.

**Considerar riesgos de seguridad.** La capa de acceso a datos debe protegerse de los ataques que intentan robar o dañar los datos. Además debe proteger los mecanismos utilizados para acceder a las fuentes de datos. También debe utilizar un enfoque de privilegios mínimos para restringir el acceso a los datos.

**Reducir la ida y vuelta.** Considere comandos "batching" o por lotes dentro de una única operación de base de datos.

**Considerar objetivos de desempeño.** Los objetivos de desempeño para el acceso a la capa de datos deben de ser tomados en cuenta durante el diseño.

## 4. Problemas comunes en la capa de acceso a datos

Existen algunos problemas comunes que se deben de considerar en el desarrollo y el diseño de la capa de acceso a datos. Estos problemas pueden ser categorizados dentro de áreas específicas del diseño. La siguiente tabla lista los problemas comunes por cada categoría:

| Categoría | Problemas comunes |
|---|---|
| **BLOB (objetos binarios grandes)** | • Usar un tipo incorrecto para los datos BLOB en la base de datos. |
| **Batching (procesamiento por lotes)** | • Ineficaz uso de procesamiento por lotes para reducir el ida y vuelta a la base de datos. |
| **Conexiones** | • Mala gestión de tiempos de espera de conexión y desconexión.<br>• Realizar operaciones que abarcan varias conexiones. |
| **Formato de datos** | • Elegir el formato de datos erróneo.<br>• No considerar los requisitos de serialización.<br>• No mapear los objetos a las fuentes de datos relacionales. |
| **Administración de excepciones** | • No controlar las excepciones de acceso a datos.<br>• No registrar excepciones importantes. |
| **Consultas (Query)** | • Mezclar consultas con la lógica del negocio.<br>• Usar cadenas de concatenación para crear consultas. |
| **Validación** | • No filtrar caracteres válidos.<br>• No manipular los valores Null. |
| **XML** | • No tener en cuenta cómo manejar conjuntos de datos XML. |

# Referencias


[1] Dissanayake Lakshman. mLearning: an innovative conceptualization to expand Education for everyone, anytime, everywhere. Leeds Metropolitan University, UK. Sprouts: Working papers on information systems. ISSN 1535-6078. 2009.

[2] Keith Clinton. Agile game development with Scrum. Pearson Education Inc. 2010. ISBN 0-321-61852-1.

[3] Martín Sergio. Claves para el desarrollo de aplicaciones móviles. IEEE TMC Spain. http://sites.ieee.org/spain-tmc

[4] Mobile Application Architecture Guide. Microsoft. (2008). Estados Unidos de América.

[5] Millard David E., Faulds Sue J, Gilbert Lester, Howard Ivonne, Sparks Dan, Wills Gary, Zhang Pei. Co-Design for conceptual spaces: an Agile design methodology por m-learning. University of Southampton, UK.

[6] Naismith, L., Lonsdale, P., Vavoula, G., and Sharples, M. Report 11: Literature review in mobile technologies and learning. In Future Series. Nesta Futures Lab, Birmingham, UK. 2005.

[7] Park Yeonjong. A Pedagogical Framework for Mobile Learning: Categorizing Educational Applications of Mobile Technologies into Four Types. The International Review of Research in Open and Distance Learning, Vol 12, No 2. 2011.

[8] Parsons D, Ryu H, Cranshaw M. A Study of Design Requirements for Mobile Learning Environments. Sixth IEEE International Conference on Advanced Learning Technologies (ICALT 2006)


Centro de Investigaciones en Computación

Instituto Tecnológico de Costa Rica

# Idea Inicial de App (IIA)

## Proyecto UbicaTEC

Cartago, Costa Rica

2013

# Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | 10/08/2013 | Jeff | |
| | | | |
| | | | |
| | | | |

# Nombre de la *app*

UbicaTEC

# Creador(es)

Equipo de Proyecto II Semestre

| Creador | Rol |
|---|---|
| Fabián Zamora | Desarrollador líder/ Programador Sénior |
| Tito  Solano | Diseñador |
| Emmanuel Mora | Desarrollador/ programador |
| Ariela Vargas | Administrador de iniciativa |

Equipo de Proyecto I Semestre 2013

| Creador | Rol |
|---|---|
| Fabián Zamora | Desarrollador líder/ Programador Sénior |
| Tito Solano | Diseñador |
| Emmanuel Mora | Desarrollador/Programador Junior |
| Ariela Vargas | Administrador de iniciativa |

Equipo de Proyecto II Semestre 2013

| Creador | Rol |
|---|---|
| Esteban Céspedes Monestel | Administrador del proyecto |
| Tito Solano | Diseñador |
| Emmanuel Mora | Desarrollador/Programador Junior |
| Fabián Zamora | Desarrollador líder/ Programador Sénior |
| Ariela Vargas | Administrador de iniciativa |

# Plataforma inicial

La aplicación estará desarrollada e implementada en el ambiente operativo android versión 2.2 y posteriores, con la futura migración al ambiente operativo IOS.

# Género

La aplicación consiste en una herramienta de geo localización dentro las instalaciones del Tecnológico de Costa Rica sede central.

# Dirigido a

Población estudiantil del Instituto Tecnológico de Costa rica.

# Estilo visual

El estilo visual de la aplicación será de 2 dimensiones.

# Concepto

La aplicación consiste en brindar una referencia precisa a los estudiantes de reciente ingreso del Instituto Tecnológico de Costa Rica sede central dentro de sus instalaciones pudiendo con esto determinar fácilmente su ubicación y obtener la información de las referencias requeridas.

Las principales características de la app son:

- Asignación automática de las referencias dentro del mapa, obtenida con base en la información de matrícula de cada estudiante suministrada por del departamento de registro y admisión.
- Asignación de actividades para cada referencia automática la cual permitirá tener un registro por parte del estudiante acerca de actividades de interés en cada curso matriculado.
- Referencia de información de miembros del curso matriculado que permitan el acceso a sus datos.

# Características

- Diseño intuitivo: La idea es crear una aplicación que sea fácil de manipular por el usuario y que muestre así la información necesaria detallando claramente las referencias solicitadas.
- Rápida autentificación: Identificación del usuario por medio el número de carnet y pin asignados.

# Aplicaciones similares

No se encontraron aplicaciones similares.

# Vistas generales



UbicaTEC
Carné
Pin
Entrar
TEC | Tecnológico de Costa Rica
3:13 PM


UbicaTEC
Entrada Principal
3:13 PM


UbicaTEC
ESCUELA DE INGENIERÍA ELECTROMECÁNICA
A-2 Escuela de Ingeniería Electromecánica
Descripción:
Lorem ipsum ad his scripta blandit partiendo, eum fastidii accumsan euripidis in, eum liber hendrerit an. Qui ut wisi vocibus suscipiantur, quo dicit ridens inciderint id. Quo mundi lobortis reformidans eu, legimus senserit definiebas an eos. Eu sit tincidunt incorrupte definitionem, vis mutat affert percipit cu, eirmod consectetuer signiferumque eu per. In usu latine equidem dolores. Quo no falli viris intellegam, ut fugit veritus placerat per. Ius id vidit volumus mandamus, vide veritus democritum te nec, ei eos debet librissulatu.
Apuntes
Marcar
3:13 PM


SONY
UbicaTEC
Carné
Pin
Entrar
TEC Tecnológico de Costa Rica
3:13 PM

Versión 1.0


Centro de Investigaciones en Computación

Instituto Tecnológico de Costa Rica

# Acta Constitutiva de Proyecto
## Proyecto UbicaTEC

Cartago, Costa Rica

2013

# Contenido

## Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | 12/08/2013 | Jeff Schmidt | |
| | | | |
| | | | |
| | | | |

# Acta de constitución del proyecto

**Fecha:** 12/08/2013

| | | |
|---|---|---|
| **Preparado por :** | | |
| Esteban Céspedes Monestel | | |
| **Revisador por :** | | |
| Jeff Schmidt | | |
| **Sponsor:** | | |
| Centro de investigaciones en computación (CIC) | | |
| **Cliente usuario final:** | | |
| Tecnológico de Costa Rica -CIC | | |
| **Tipo de proyecto:** | | |
| Informático-aplicación móvil | | |
| **Desarrollador líder:** | | |
| Fabián Zamora | | |
| **Unidades de negocio:** | | |
| Usuarios con dispositivos móviles compatibles con el producto desarrollado. | | |
| Sponsor que autoriza el proyecto | | |
| **Nombre** | **Firma** | **Fecha** |
| | | |

## Racionalidad y Propósito del Proyecto

La apertura de nuevas tecnologías en pleno siglo XXI ha permitido el uso de herramientas informáticas que han facilitado las actividades cotidianas del ser humano, muchas de estas herramientas son gratuitas y de fácil acceso permitiendo así su uso y aprovechamiento. En particular el uso de la red global de posicional conocida como GPS, la intuye al usuario de la misma acerca de las locaciones y ubicaciones cercanas, permitiendo así el ahorro del recurso más importante que se tiene hoy en día el tiempo.

## Justificación del Proyecto

Una de las necesidades que afronta el estudiante de primer ingreso del Instituto Tecnológico de Costa rica dentro de sus instalaciones sede central Cartago, es la falta de información precisa acerca de sus instalaciones y referencias. El proyecto UbicaTEC buscará satisfacer esa necesidad poniendo a disposición de la población estudiantil una app que permitirá determinar con precisión la ubicación del usuario así como guiarlo en forma efectiva durante su estancia dentro de las instalaciones del Instituto Tecnológico de Costa Rica sede central Cartago.

## Metas y objetivos

El Proyecto tendrá como meta generar una aplicación en la plataforma Android. Los objetivos del proyecto son los siguientes:

- Diseñar y desarrollar una aplicación de posicionamiento que permita de manera intuitiva al usuario su correcta ubicación dentro de las instalaciones del Tecnológico de Costa rica.
- Añadir elementos a la aplicación que permitan la administración de actividades dentro de cada referencia mostrada en el mapa.

## Estrategia del Proyecto

El equipo del proyecto UbicaTEC consolidara el éxito del proyecto basado en la disposición y contribución efectiva del talento humano designado, para ello se dispondrá de 5 horas semanales en las cuales cada miembro del equipo tendrá una asignación de laborales a las cuales se realizara un seguimiento continuo.

## Estructura de gobernabilidad

El directorio de supervisión del proyecto está compuesto por Director del proyecto (PM por sus siglas en inglés) quien desempeñara la función de coordinación del proyecto, Desarrollador líder del proyecto quien implementara los diseños y funcionalidades definidas por medio de su programación dentro de la aplicación , Desarrollador adjunto quien brindara la asistencia necesaria al desarrollador líder, Diseñador de la aplicación encargado de definir los diseño necesario de la aplicación basado en los requisitos definidos.

## Alcance del proyecto

El Desarrollo de una aplicación de geo localización, así como las pruebas necesarias para su correcto desempeño y gestión de las expectativas del usuario final.

## Dependencias y links con otros proyectos

El proyecto parte de la iniciativa y apoyo del proyecto de investigación Arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles.

## Factores críticos de éxito

- Compromiso de los miembros en la realización de sus funciones.
- Apoyo constante por parte de los stakeholders.

## Supuestos

- El equipo podrá ser uso de las instalaciones designadas dentro del CIC para la gestión de labores.
- Se brindarán las herramientas necesarias para las pruebas de la aplicación.
- El equipo del proyecto estará en constante comunicación a través de internet.

# Requerimientos de personal

| Puesto/rol | Principal interés en el proyecto | Otras responsabilidades |
|---|---|---|
| Administrador de iniciativa | Solicitar al administrador del proyecto y/o avances acerca del estado del proyecto | Apoyar por medio de Juicio experto. |
| Administrador del proyecto | Velar por el cumplimiento del alcance y objetivo del proyecto. | Fortalecer la relación y comunicación del equipo del proyecto. |
| Desarrollador líder | Velar porque los requerimientos solicitados sean correctamente implementados y las expectativas adjuntas a estos sean cumplidas dentro de los estándares definidos. | Encargado de la definición técnica, brindar opiniones acerca del diseño de funcionalidades, coordinar las tareas con el diseñador a su cargo. |
| Desarrollador | Apoyar al desarrollador líder en el cumplimento de objetivos. | Apoyar al equipo de proyecto brindado opiniones. |
| Diseñador | Encargado de la definición visual de la aplicación así como la definición de su interacción. | Brindar apoyo constante al desarrollador líder por medio de los diseños solicitados. |

| Recursos requeridos | Responsable |
|---|---|
| Lugar físico de trabajo | Para tal actividad el lugar será dependiente a la conveniencia de cada miembro del equipo para la realización de sus labores, en el caso de las reuniones están se llevaran a cabo dentro en la instalaciones del centro de investigación en computación (CIC). |
| Repositorio centralizado de código fuente | Para este efecto se estará implementando el servicio git hub. |

# Hitos principales del proyecto

| Cronograma hitos del proyecto | |
|---|---|
| Hito o evento significativo | Fecha programada |
| Inicio Proyecto II semestre | 01-agosto-2013 |
| Entrega de informes | 09-septiembre-2013 |
| Entrega del acta de constitución del proyecto | 13-septiembre-2013 |
| Entrega del Documento de Idea Inicial de Aplicación | 13-septiembre-2012 |
| Entrega del Documento de Diseño de la Aplicación | |


## Funciones

1. Inicio de sesión:
Para estudiantes, por medio de carnet y pin

2. Sincronización con el horario matriculado, incluyendo: Información del curso, compañeros, ubicación en el mapa, y posibilidad de escribir notas.

3. Mapa interactivo del TEC:
Muestra la ubicación del usuario por medio de GPS, contiene los edificios más importantes, su información y diferentes herramientas.

4. Permite el acceso a el mapa a usuarios sin iniciar sesión.

gotouch

GoTouch nace en el año 2011 como una iniciativa del Centro de Investigaciones en Computación (CIC), el cuál pertenece al Instituto Tecnológico de Costa Rica (TEC).

**Objetivo**
Desarrollar juegos y aplicaciones educativas para las plataformas móviles iOS y Android, que
colaboren en la promoción a nivel nacional e internacional de la Escuela de Computación del
TEC, el CIC y el TEC.

**Coordinador de Proyecto:**
Prof. Jeff Schmidt Peralta

**Lider Técnico:**
Prof. Andrei Fuentes Leiva

**Directora de Proyecto:**
Ariela Vargas Vargas

# UbicaTEC

# UbicaTEC

Es una aplicación para Android, creada para facilitar la ubicación de los diferentes edificios, y servicios, tanto a las personas que visitan la institución como para estudiantes, brindando a estos ultimos, diferentes herramientas para organizar las tareas que realizan dentro del campus del Tecnológico de Costa Rica.



**Desarrollo:** Fabian Zamora Ramírez
**Diseño:** Tito Solano Villaobos


Centro de Investigaciones en Computación

Instituto Tecnológico de Costa Rica

# Idea Inicial de App (IIA)

## Proyecto de Operaciones Básicas

Cartago, Costa Rica

2013

# Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | 09/09/2013 | Jeff Schmidt | |
| 02 | 10/09/2013 | | |
| | | | |
| | | | |

# Nombre de la app

+Plus

# Creador(es)

Equipo de Proyecto II Semestre 2012

| Creador | Rol |
|---|---|
| Natalia Quirós | Programadora |
| Andrés Morales | Programador |
| Elia Estrada | Diseñadora |
| Paula Hidalgo | Diseñadora |

Equipo de Proyecto I Semestre 2013

| Creador | Rol |
|---|---|
| Herberth Torres | Administrador de Proyecto |
| Andrés Morales | Programador Sénior |
| Álvaro Fallas | Programador Junior |
| Alejandro Soto | Programador Junior |
| Jorge Arguedas | Programador Junior |

# Plataforma inicial

En primera instancia se desarrollará en plataforma Android 2.3.3, y posteriormente para alcanzar el mayor público posible se considerará extender la aplicación con una versión para iOS.

# Género

La aplicación consiste en un juego educativo.

# Dirigido a

El juego está orientado a niños entre 4 y 7 años de edad.

# Estilo visual

El estilo visual de la aplicación será de 2 dimensiones.

# Concepto

El juego consiste en 3 mundos:

- El primero es pensado para que el niño conozca los números y se familiarice con estos, además de aprender a contar y asociar los números con las cantidades que representan.
- El segundo mundo es para enseñarle al niño el concepto de las sumas a través de dos niveles, uno en el que se realizan las sumas contando estrellas y un juego en el que se revientan burbujas con las sumas y los resultados.
- El tercer mundo consiste en enseñarle al niño la manera adecuada de dibujar los números aprovechando las facilidades que otorgan los dispositivos móviles para detectar los trazos dibujados.

# Características

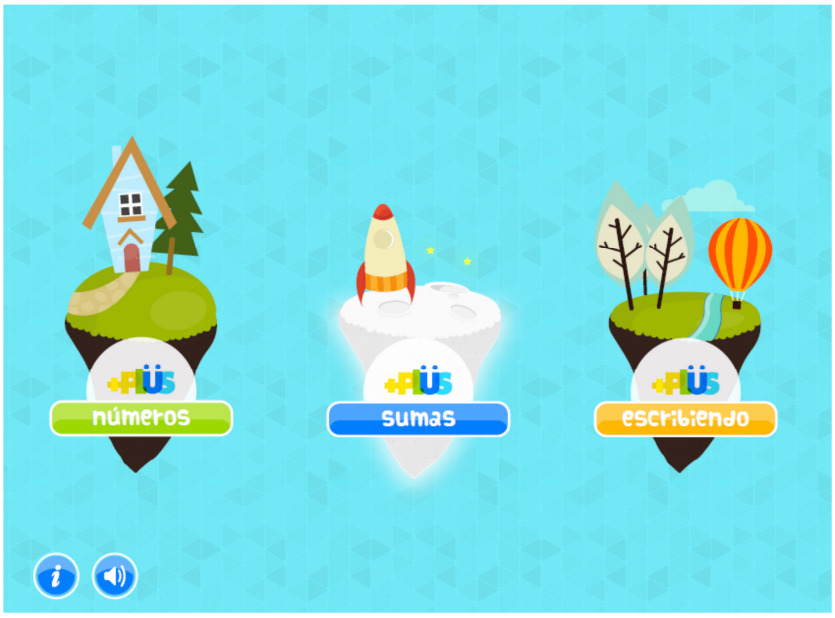
Jugabilidad sencilla: La idea de cómo se desarrolla el juego debe de ser fácil de entender, de manera que no sea un reto para el usuario entender de qué se trata la dinámica del juego.

# Aplicaciones similares

No se encontraron aplicaciones similares.

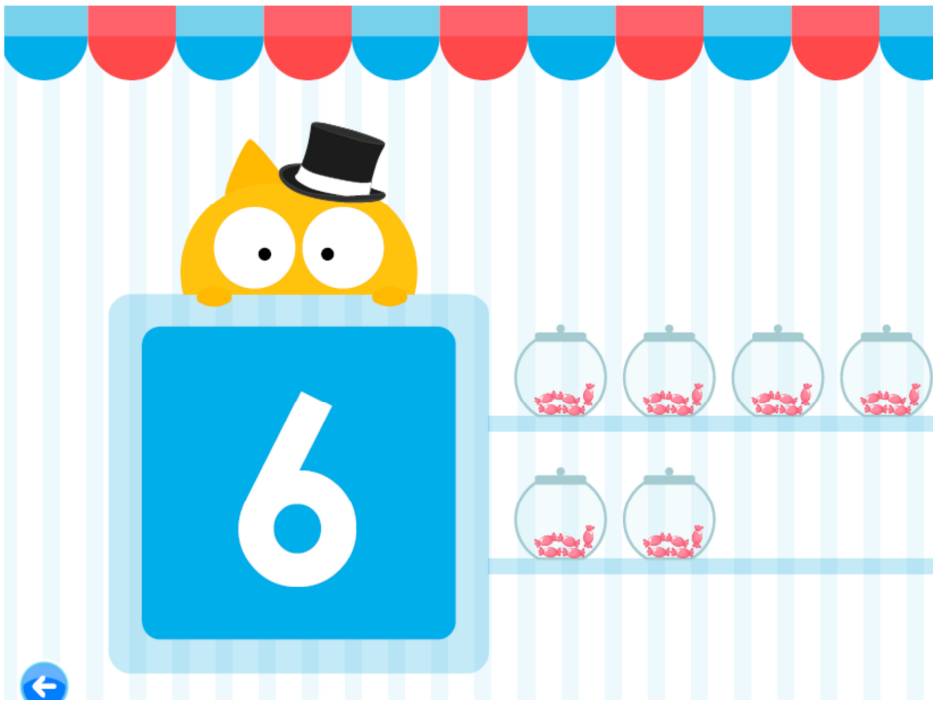
# Vistas generales

Menú Principal

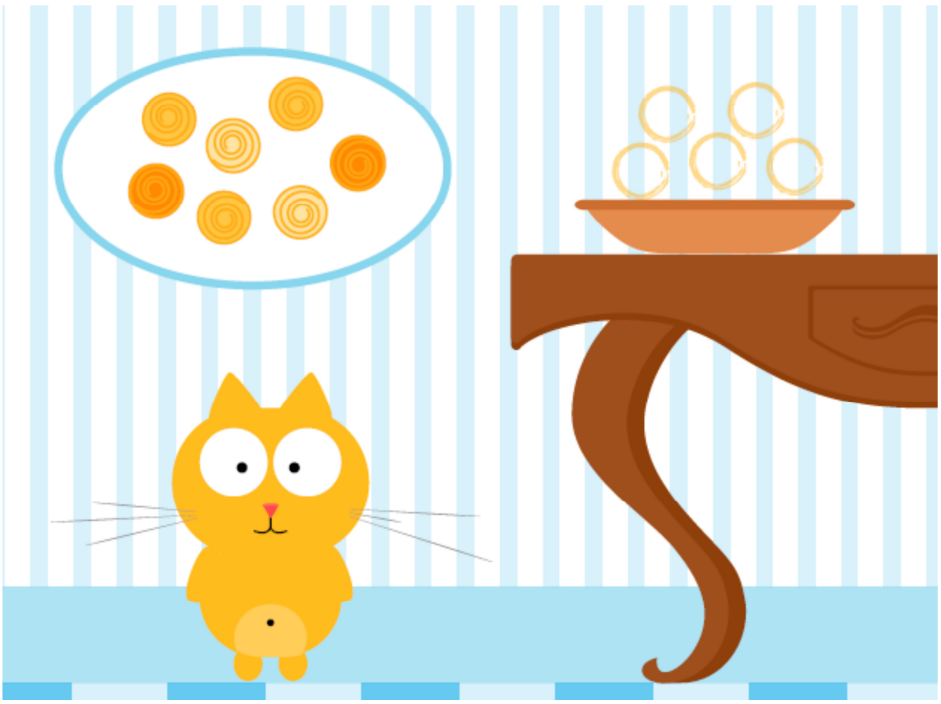



Menú del Mundo 1

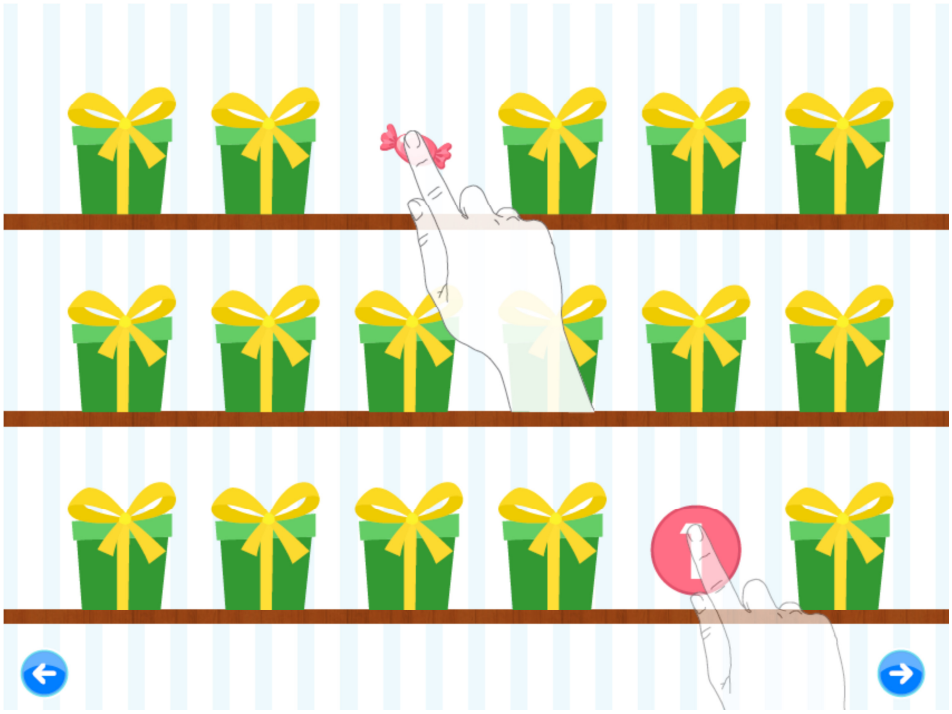
Ejemplo de Nivel de Conocer los Números



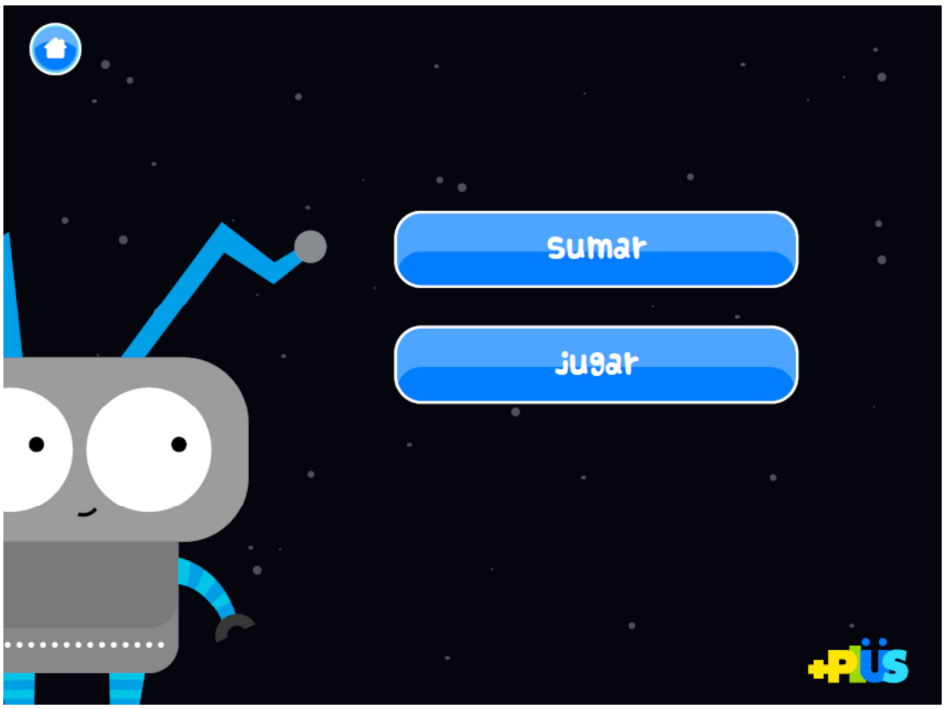

Ejemplo Nivel de Contar



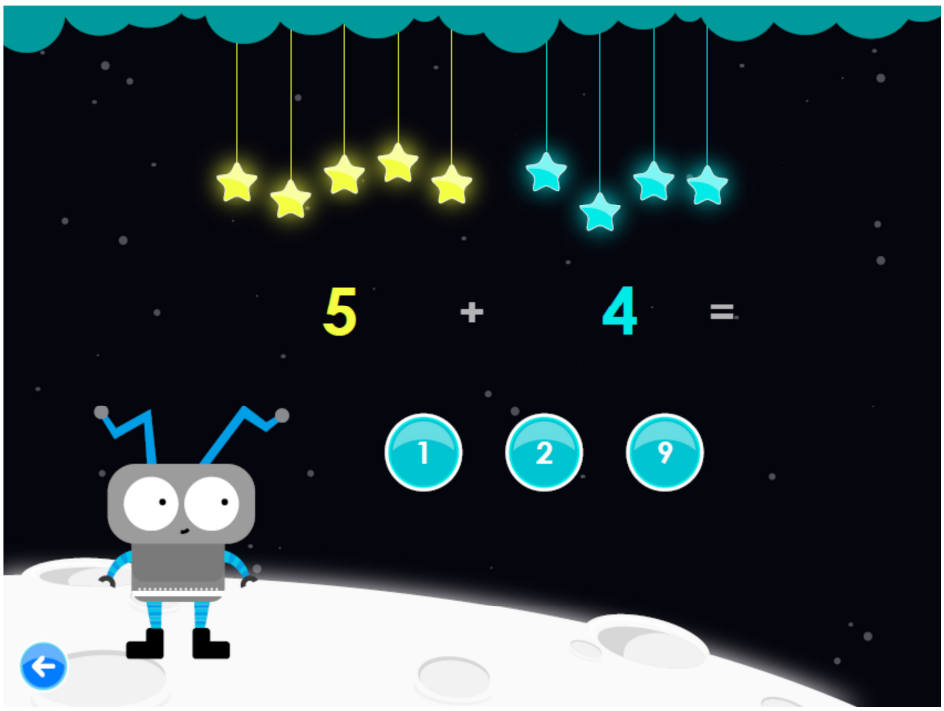
Ejemplo Ejercicio de Jugar  a las Parejas



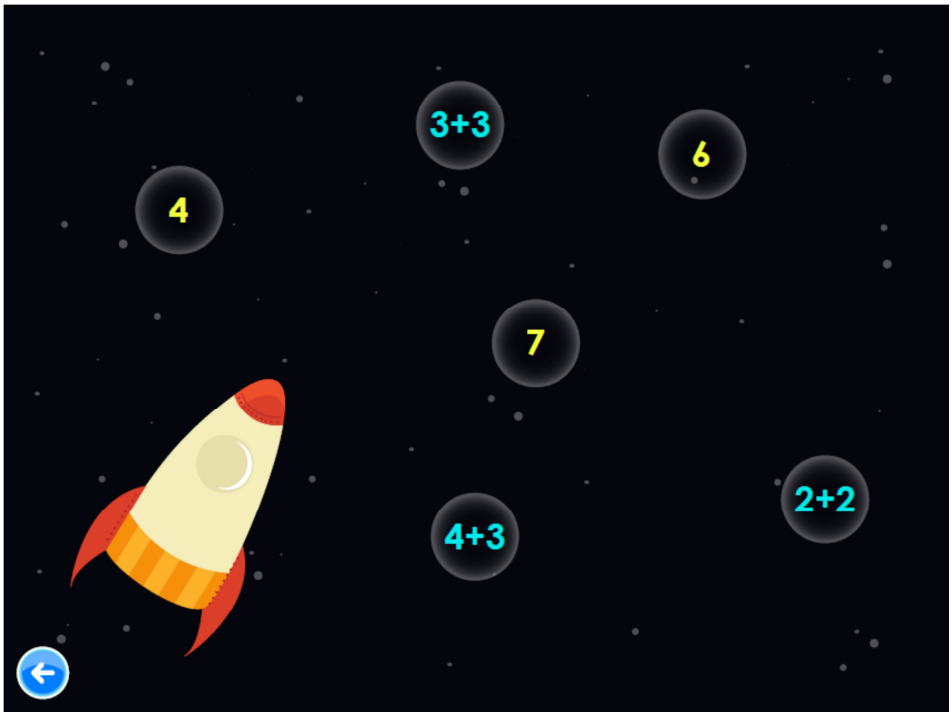
Menú del Mundo 2



Ejemplo del Nivel de Sumas




Ejemplo del Nivel de Jugar de Explotar Parejas



Ejemplo del Nivel de Dibujar los Números


# Referencias

No hay ninguna fuente en el documento actual.


Centro de Investigaciones en Computación

Instituto Tecnológico de Costa Rica

# Acta Constitutiva de Proyecto (ACP)

## Proyecto de Operaciones Básicas

Cartago, Costa Rica

2013

# Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | 10/09/2013 | Jeff Schmidt | |
| | | | |
| | | | |
| | | | |

# Acta de constitución del proyecto

**Fecha:** 13/03/2013
**Versión:** 01

| | |
|---|---|
| **Preparado por** | Herberth Torres |
| **Revisado por** | Jeff Schmidt |
| **Sponsor** | Centro de Investigaciones en Computación (CIC) |
| **Cliente/Usuario final** | ITCR – CIC |
| **Contacto del cliente** | María Marta Camacho - UCR |
| **Tipo de proyecto** | Productivo – Informático |
| **Desarrollador líder** | Andrés Morales |
| **Unidades de negocio** | Clientes con dispositivos móviles compatibles con los productos generados. |
| **Firma del responsable** | |

# Visión general del proyecto

## Descripción del proyecto

El juego consiste en 3 mundos, el primero es pensado para que el niño conozca los números y se familiarice con estos, además de aprender a contar y asociar los números con las cantidades que representan.

El segundo mundo es para enseñarle al niño el concepto de las sumas a través de dos niveles, uno en el que se realizan las sumas contando estrellas y un juego en el que se revientan burbujas con las sumas y los resultados.

El tercer mundo consiste en enseñarle al niño la manera adecuada de dibujar los números aprovechando las facilidades que otorgan los dispositivos móviles para detectar los trazos dibujados.

## Justificación del Proyecto

Lo que motiva este proyecto es la posibilidad de familiarizar a niños y con los conceptos matemáticos básicos a edades tempranas de una manera divertida en interactiva. Lo que se busca es eliminar la barrera de que la matemática es una materia difícil haciendo que los conceptos matemáticos sean percibidos intuitivamente a través de un juego.

La primera fase del proyecto se centra en conocer los números, contar, sumar y escribir los números, pero posteriormente se extenderá para incluir otros conceptos matemáticos como las otras operaciones básicas (resta, multiplicación, división).

## Metas y objetivos

El Proyecto tendrá como meta desarrollar una aplicación para dispositivos móviles que desarrolle conceptos matemáticas básicos de una manera interactiva y sencilla y le facilite al usuario la comprensión de estos conceptos.

Los objetivos del proyecto son los siguientes:

- Desarrollar un juego en Android que abarque el conocimiento de los números, contar, sumar y la escritura de los números.
- Cubrir el alcance esperado, en el cronograma acordado, y según el presupuesto asignado, siempre manteniendo un alto nivel de calidad en los entregables.

# Alcance del proyecto

### El proyecto incluye

Desarrollo de un juego con 3 escenarios, el primer release abarcará los conceptos de conocer los número y contar (mundo 1), sumas (mundo 2) y escritura de los números (mundo 3).

### El proyecto no incluye

Este primer release no incluirá el desarrollo de las otras operaciones básicas (resta, multiplicación o división).

| Entregable | Criterios de aceptación |
|---|---|
| IIA | Cumplir con la plantilla dispuesta para este fin. |
| ACP | Cumplir con la plantilla dispuesta para este fin. |
| DDA | Cumplir con la plantilla dispuesta para este fin. |
| Aplicación | Se entregará el código fuente y todo lo necesario para continuar con el proyecto en semestres posteriores.<br>Se debe ajustar al estándar de desarrollo de apps de GoTouch. |
| Diseño de la aplicación | Se entregarán los archivos editables y las imágenes finales y todo lo necesario para continuar con el proyecto en semestres posteriores.<br>Se debe ajustar al estándar de diseño de apps de GoTouch. |

### Dependencias y links con otros proyectos

Este proyecto depende del proyecto de investigación Arquitectura para el desarrollo de aplicaciones educativas para dispositivos móviles dentro del Centro de Investigaciones en Computación (CIC) del TEC.

Es el primer proyecto de este tipo en la iniciativa, pero a partir de este se podrán generar proyectos para cubrir releases posteriores.

# Factores críticos de éxito

- El equipo de proyecto debe estar comprometido y dedicarle las horas que tienen asignadas al proyecto.
- El proyecto debe tener el apoyo del stakeholder María Marta Camacho.

## Supuestos

- El equipo de proyecto estará dispuesto a utilizar sus recursos para la realización del proyecto, en caso de algún contratiempo podrá utilizarse activos disponibles de GoTouch para la realización de los trabajos.

# Requerimientos de personal

| **Puesto/Rol** | **Principal interés en el proyecto** | **Otras responsabilidades** |
|---|---|---|
| Administrador de iniciativa | Solicitar al administrador de proyecto avances y/o informes del proyecto | |
| Administrador del Proyecto | Coordinar las actividades para llevar el proyecto a un desarrollo exitoso. | Generar la documentación del proyecto, coordinar reuniones y ser responsable por el proyecto. Facilitar la obtención de patrocinio de empresas terceras y representar al equipo desarrollador ante la VIE, CIC y TEC. |
| Desarrollador líder | El proyecto sea terminado con las expectativas y necesidades requeridas | Guiar el trabajo del equipo desarrollador, formular ideas y propuestas, coordinar y enseñarles a los otros desarrolladores. |
| Programador | Realizas las tareas de desarrollo de software que le son asignadas, colaborar y dar ideas a lo largo del proyecto. | |
| Diseñador gráfico | El papel del diseñador corresponde a casi el 50% de la importancia de una aplicación de este tipo, el diseñador genera las vistas de la aplicación y se preocupa por la usabilidad de la aplicación. | Pedir comentarios y sugerencias a los demás miembros del equipo |

# Instalaciones y recursos del proyecto

| **Recursos requeridos** | **Responsable** |
|---|---|
| Ambiente de desarrollo especializado con NME y FlashDevelop | Cada uno de los desarrolladores aporta su máquina. |
| Lugar físico de trabajo. | Por el tipo de proyecto cada miembro trabaja en el lugar de su preferencia, las reuniones serán en la oficina de GoTouch en el CIC. |
| Repositorio centralizado de código fuente | Se utilizará un repositorio en Github. |

# Hitos principales del proyecto

| Hito | Fecha |
|---|---|
| Entrega del Documento de Idea Inicial de Aplicación | 14/09/2013 |
| Entrega del Acta de Constitución de Proyecto. | |
| Entrega del Documento de Diseño de la Aplicación | |
| Entrega del Desarrollo del Semestre | |
| Entrega del Diseño del Semestre | |


Centro de Investigaciones en Computación

Instituto Tecnológico de Costa Rica

# Metodología de desarrollo de Operaciones Básicas

Autores:

Jeff Schmidt Peralta

Aldo Mora Fernández

Cartago, Costa Rica

2013

# Contenido

# Historial de las revisiones

| Versión | Fecha | Revisado por | Descripción/Estado |
|---|---|---|---|
| 01 | 5/09/2013 | Jeff Schmidt | |
| | | | |
| | | | |
| | | | |

# Introducción

En el año 2011, se decide crear en el TEC una iniciativa para agrupar proyectos relacionados con realizar aplicaciones y juegos, que brinden soluciones innovadoras a necesidades existentes, que sirvan como apoyo al sistema educativo formal y que en igual medida impulsen el aprendizaje autodidacta en aquellas poblaciones que posean acceso a dispositivos móviles, inicialmente con sistemas operativos iOS o Android.

La iniciativa se llama GoTouch y se establece formalmente a partir de setiembre de 2011 en el Centro de Investigaciones en Computación en el TEC. La iniciativa está conformada por profesores de la Escuela de Computación del TEC, además de estudiantes de diversas carreras del TEC. El objetivo general que persigue la iniciativa es desarrollar juegos y aplicaciones educativas para dispositivos móviles.

Para cumplir con este objetivo se piensa en el desarrollo de un proyecto que brinde la posibilidad de familiarizar a niños y con los conceptos matemáticos básicos a edades tempranas de una manera divertida en interactiva. Lo que se busca es eliminar la barrera de que la matemática haciendo que los conceptos sean percibidos intuitivamente a través de un juego.

# Las metodologías de desarrollo

Una metodología de desarrollo es el conjunto de procedimientos, técnicas, documentos y ayudas que se van a realizar para la construcción de software, en nuestro caso de apps para dispositivos móviles.

Una app es una aplicación nativa específica para un dispositivo móvil para ser ejecutada en una plataforma móvil. Normalmente son descargadas a través de Internet desde sitios y cargadas directamente en el dispositivo. Las apps pueden tomar ventaja de los características de los dispositivos móviles, tales como cámaras, geolocalización, micrófonos, parlantes, sensores de movimiento y otros.

Las metodologías más utilizadas en la actualidad, han sido desarrolladas basándose en el ciclo de vida de los sistemas y pensando en proyectos de software tradicionales. El ciclo de vida comprende las fases y que normalmente van desde las etapas de análisis y especificación de requerimientos, diseño, programación, pruebas e implementación.

Estas metodologías para el desarrollo de sistemas han estado adaptándose a la producción de software para dispositivos móviles. Algunas de las metodologías más utilizadas son:

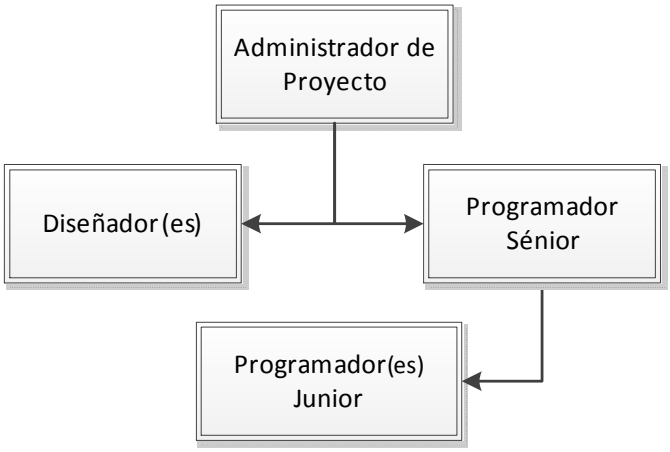
- Cascada (Waterfall): consiste en dividir el proyecto en fases secuenciales, basadas en el ciclo de vida de los sistemas. El desarrollo se realiza en fases, dando énfasis al planeamiento, cronogramas, presupuestos e implementación del sistema.
- Ágil (Agile): está basada en el desarrollo iterativo, en el cual todas las fases del ciclo del proyecto son desglosadas en partes más pequeñas. El aspecto principal son los requerimientos.
- Programación Extrema (Extreme Programming): XP pretende conseguir la calidad del software ante los cambiantes requerimientos del usuario. Como un tipo de desarrollo ágil se intentan tener múltiples ciclos cortos de desarrollo, con la finalidad de reducir el costo de cambios.
- Desarrollo de Rápida Acción (Rapid Action Development): RAD se basa en evitar planeación excesiva para proyectos urgentes, de forma que sea fácil adaptarse a los cambios en los requerimientos.
- Proceso Scrum: es un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto.

Las empresas que desarrollan apps han adoptado algunas de estas metodologías a los nuevos desafíos del desarrollo en la computación móvil, sin que existan a la fecha estándares en la industria.

La iniciativa GoTouch pretende desarrollar especialmente apps educativas. El aprendizaje móvil (m-learning) permite que el acceso al conocimiento pueda darse en el momento adecuado, ya que la instrucción puede realizarse en cualquier momento y lugar, utilizando un dispositivo móvil. Bajo diversos paradigmas de la teoría del aprendizaje pueden construirse aplicaciones que apoyen y complementen los sistemas de educación.

# Equipo de trabajo

El equipo de trabajo encargado de un proyecto puede apreciarse en la siguiente figura:




Los roles dentro del proyecto se describen a continuación:

## Administrador de Proyecto

El administrador de la iniciativa es el responsable del control del proyecto. En la fase de exploración e iniciación debe elaborar el Acta Constitutiva del Proyecto (ACP), documento que define de manera general la app a desarrollar, recursos requeridos y los principales hitos que van a regir el desarrollo.

## Programador Sénior

El desarrollador líder es el encargado de liderar el equipo de desarrollo de una app. Participa en todas las fases del desarrollo y es el responsable de hacer el mejor uso de los recursos que se asignen.

## Diseñadores

Los diseñadores participan activamente en el proyecto desde sus fases iniciales y son los encargados de proponer y desarrollar el entorno gráfico de la app, así como las diferentes alternativas de interacción humano – dispositivo móvil.

## Programadores Junior

Los programadores junto con el desarrollador líder son los encargados de codificar los diferentes componentes de software que van a constituir la app. Además van a participar en las etapas de testing y lanzamiento.

## Composición del Equipo

Equipo de Proyecto II Semestre 2012

| Creador | Rol |
|---|---|
| Natalia Quirós | Programadora |
| Andrés Morales | Programador |
| Elia Estrada | Diseñadora |
| Paula Hidalgo | Diseñadora |

Equipo de Proyecto I Semestre 2013

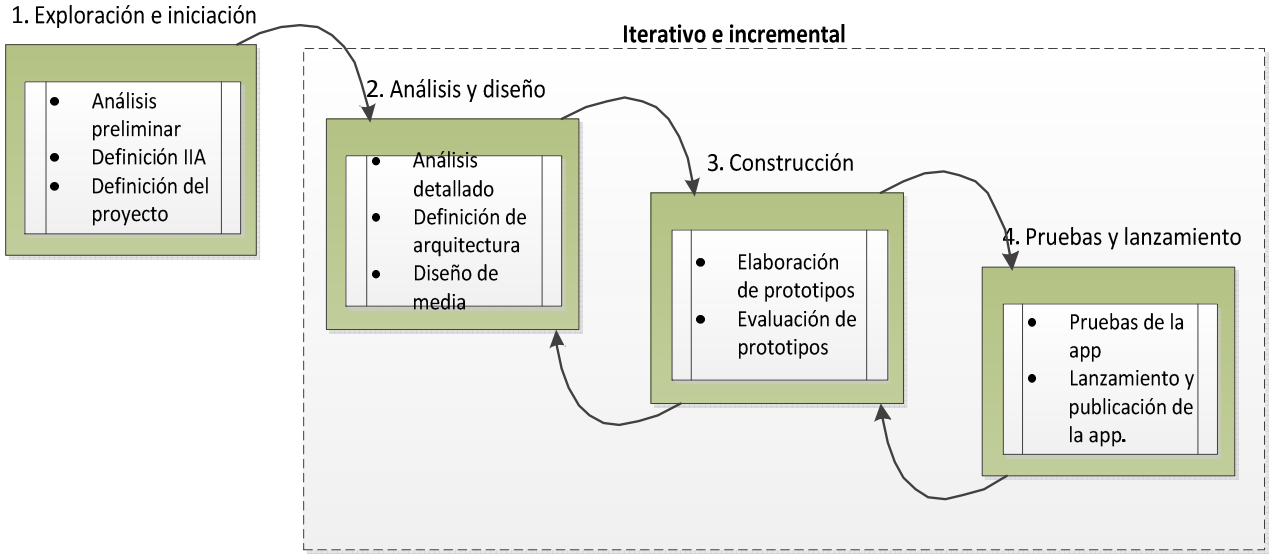
| Creador | Rol |
|---|---|
| Herberth Torres | Administrador de Proyecto |
| Andrés Morales | Programador Sénior |
| Álvaro Fallas | Programador Junior |
| Alejandro Soto | Programador Junior |
| Jorge Arguedas | Programador Junior |

# Descripción general de la metodología

La metodología a utilizar para los proyectos de GoTouch utiliza conceptos y buenas prácticas de las metodologías descritas anteriormente, basándose en los conceptos de un desarrollo iterativo e incremental.

Se hace énfasis en los resultados, para obtener un producto de calidad en el menor tiempo posible. La documentación inherente al proyecto no es exhaustiva, pero si la necesaria para poder darle continuidad.

Las fases del ciclo de vida de una app en esta metodología, pueden apreciarse en la siguiente figura:




Después de la primera fase en la cual se inicia el proceso de desarrollo, las tres fases siguientes deben ser iterativas. Las iteraciones deben ser pequeñas, de una a dos semanas, que permitan llevar el control del avance en el desarrollo.

Las etapas de las fases 2, 3 y 4 no son secuenciales, pueden traslaparse en el tiempo y pueden ser desarrolladas en paralelo.

## FASE 1. Exploración e iniciación

Esta fase se inicia con una idea inicial para una app. Esta idea debe ser analizada y en caso de determinarse su factibilidad a nivel técnico y de recursos, se documenta esta etapa inicial.

La fase de exploración e iniciación consta de las siguientes etapas:

## Análisis preliminar

En esta etapa se analiza la idea inicial de la app. Los miembros del equipo de desarrollo deben discutir el alcance que se desea alcanzar y desarrollar en forma paralela el documento de Idea Inicial de App.

## Definición de la idea inicial de la app

En esta primera etapa se debe definir de manera muy general la app a desarrollar. Se debe utilizar el documento Idea Inicial de App o IIA. El tiempo normal para esta etapa debería ser entre 1 y 2 semanas.

## Definición del proyecto

Una vez realizado el documento IIA, el administrador de la iniciativa es el responsable de elaborar una Acta Constitutiva del Proyecto (ACP), que permita conocer la visión del proyecto, los alcances y requerimientos de personal y otros recursos que se van a utilizar para poder desarrollarlo. Además se deben definir los hitos principales del proyecto. La duración de la elaboración del ACP no debe ser mayor a 1 semana.

# FASE 2. Análisis y diseño

En esta etapa se define toda la información que va a regir el desarrollo de la app. El producto de esta fase se debe resumir en el Documento de Diseño de App o DDA. Las etapas son iterativas, los cambios en cualquier parte del proceso deben documentarse en el DDA, el cual no debe verse como un documento estático, sino como la base para documentar este proceso iterativo.

El modelo se basa en realizar iteraciones o sprints semanales o quincenales, que permitan conocer los avances del proyecto. En estos puntos de control el desarrollador líder se reunirá con el administrador de la iniciativa, para conocer los avances en el proceso.

Las etapas de esta fase son:

### Análisis detallado.

En esta etapa se deben definir los objetivos y el alcance de la app, en forma incremental, se inicia con ideas generales que se pueden ir aplicando en la Fase 3 Construcción, de tal forma que se vayan generando prototipos.

### Definición de arquitectura.

Conforme se van conociendo los alcances de la app, se debe ir diseñando la arquitectura que va a tener la app, considerando la plataforma para la cual se va a desarrollar. La arquitectura puede cambiar según el avance en el proceso de desarrollo.

### Diseño de media.

Los aspectos relacionados con la presentación de la app y la interfaz del usuario con la misma deben diseñarse en esta etapa. Se puede iniciar con bocetos o mockups, los cuales una vez aprobados por el equipo de trabajo pueden irse desarrollando formalmente.

# FASE 3. Construcción

En esta fase se construyen los componentes de software que van a conformar la app, considerando la utilización de los repositorios de código de la iniciativa. La codificación va dirigida a la elaboración de prototipos incrementales, a los cuales en cada sprint se le van agregando funcionalidades.

Las etapas de la fase de construcción son:

### Elaboración de prototipos.

Conforme se avanza el análisis y diseño de la app, se deben ir construyendo los prototipos que van a ser evaluados en cada Sprint.

### Evaluación de prototipos.

El equipo de desarrollo debe evaluar cada prototipo desarrollado, definiendo que aspectos deben mejorarse y cuales van a ser los aspectos a incluir para el próximo sprint.

# FASE 4. Pruebas y lanzamiento

En esta fase se realiza el deployment de la app, se inician las pruebas hasta lograr un estado aceptable que permita obtener una primer reléase que pueda ser publicado.

Las etapas de esta fase de pruebas y lanzamiento son:

## Pruebas de la app.

El testing de la app debe realizarse conforme se van liberando los prototipos, por parte de los miembros del equipo. Es importante probar diferentes tipos de dispositivos para asegurar la compatibilidad de la app.

Una vez que se ha conseguido una versión estable y que cumpla con los requerimientos definidos, es importante hacer una evaluación con una muestra de la población meta a la cual va dirigida la app, con la finalidad de obtener retroalimentación.

## Lanzamiento y publicación de la app.

En la etapa de lanzamiento se debe definir la estrategia que va a regir la publicación de la app, considerando aspectos como divulgación, si se va a cobrar o no y en que sitios se va a promover.

# Descripción general de la metodología aplicada a Operaciones Básicas

El proyecto se desarrolló bajo una metodología denominada Scrum adoptado, que consiste en un proceso de Scrum normal en lo que respecta a las entregas rápidas, pero se varía la duración de los sprints. La duración de los mismos se definía mediante medios electrónicos o en las reuniones realizadas de forma semanal o quincenal dependiendo de la disponibilidad del equipo de trabajo.

En lo que respecta a las reuniones se definían siempre ciertos aspectos claves para el proyecto como: el estado actual del proyecto, los incidentes ocurridos durante la semana y la definición de “next steps” en otras palabras los pasos que se deben realizar para proseguir con el desarrollo del proyecto.

# Referencias

Keith, Clinton. Agile game development with Scrum. Addison Wesley, 2010.

Martín Sergio. Claves para el desarrollo de aplicaciones móviles. IEEE TMC Spain. http://sites.ieee.org/spain-tmc

Naismith, L., Lonsdale, P., Vavoula, G., and Sharples, M. Report 11: Literature

review in mobile technologies and learning. In Future Series. Nesta Futures Lab,

Birmingham, UK. 2005.

# +plus
guía de juego

# 1

## nivel 1

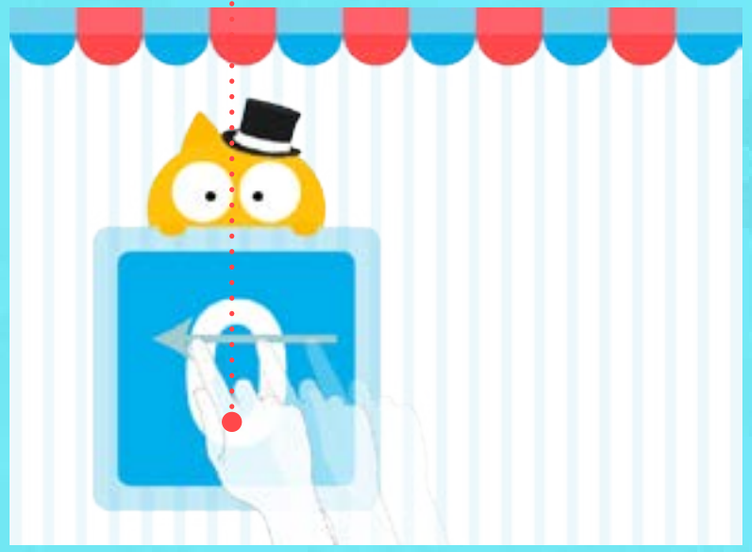
Objetivo. conocer los números y aprender a contar, relacionando números con numerales.

# nivel 1

1.1. “conoce los números”

El objetivo de este apartado es que el niño se familiarice con los numerales y los asocie con su respectiva cantidad o conjunto.

el niño podrá deslizar la pantalla e irán apareciendo los diferentes números con su respectivo conjunto.



# nivel 1

1.2. “contar”

En esta parte del juego el personaje propone un reto, que será una cantidad de ovillos para arrastrar.

El niño debe arrastrar los ovillos, mientras el narrador los va contando.

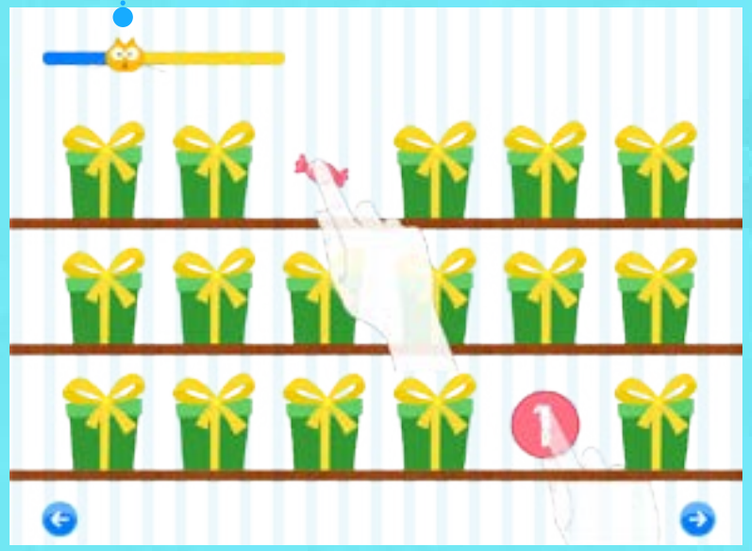
# nivel 1

1.3. “jugar”

El objetivo de esta última parte es que el niño aplique lo aprendido jugando.

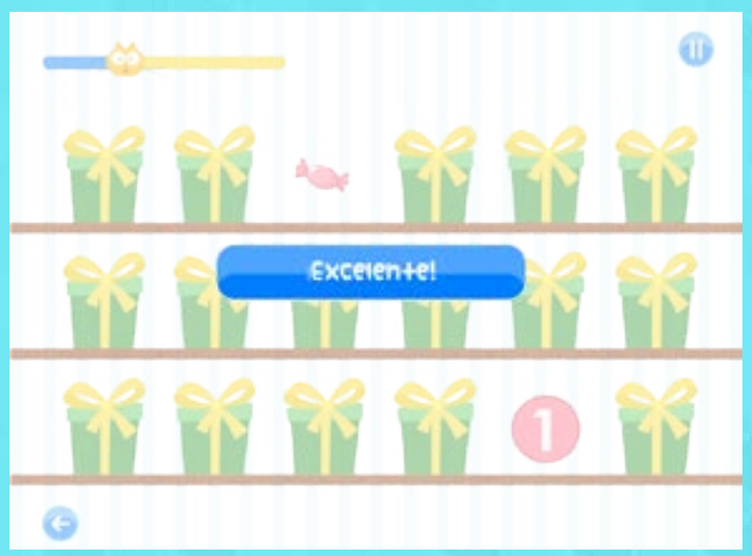
Consiste en un juego de memoria donde se debe asociar un conjunto con su respectivo numeral.

barra de tiempo

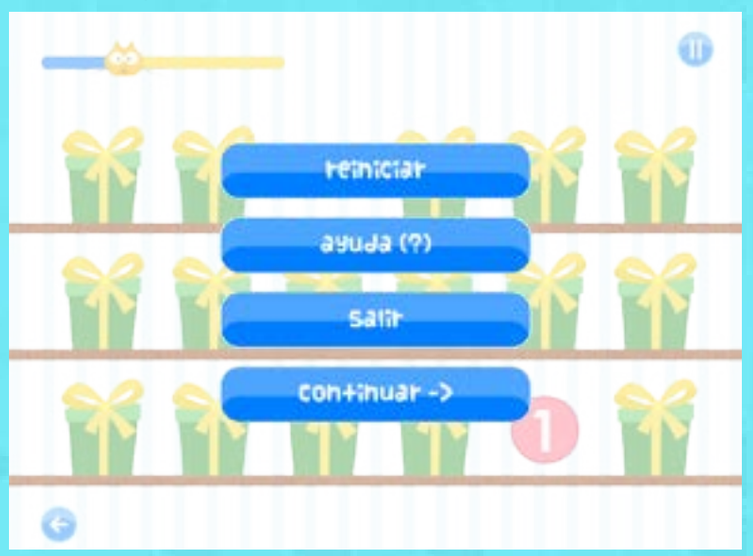


pantalla al acabarse el tiempo y perder


pantalla al ganar




menú de pausa



2

# nivel 2

Objetivo. aprender a sumar relacionando cantidades con numerales.

# nivel 2

2.1. “sumar”

**¿Cómo jugar?**

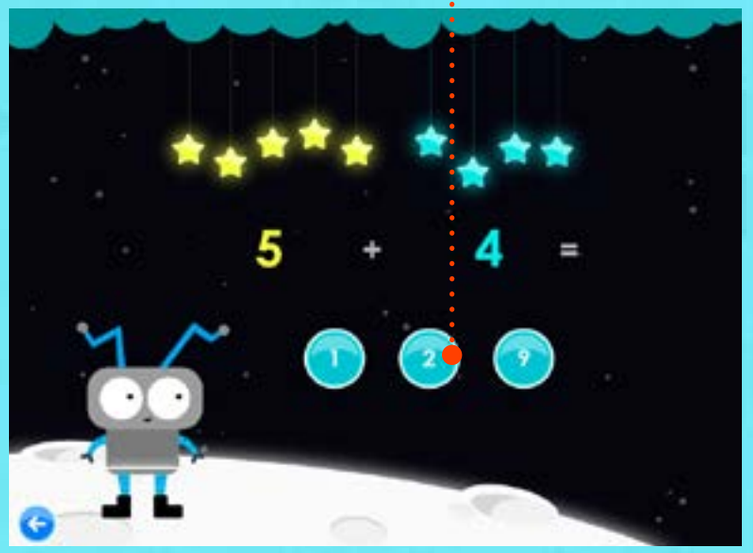
1. Aparecen los numerales. El niño debe arrastrar la cantidad de estrellas asociada al numeral, según el color de cada uno.

2. El niño debe realizar la suma, seleccionando entre un grupo de respuestas.

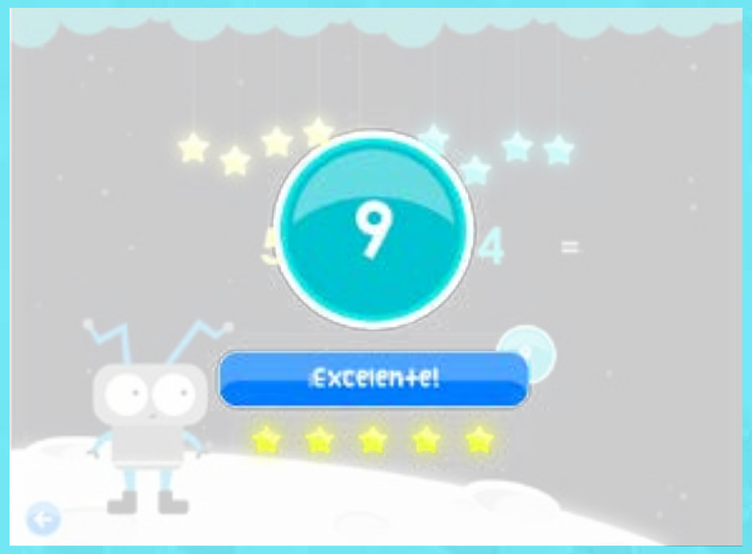
3. Se darán cinco oportunidades, por cada respuesta correcta se otorgará una estrella.

1.

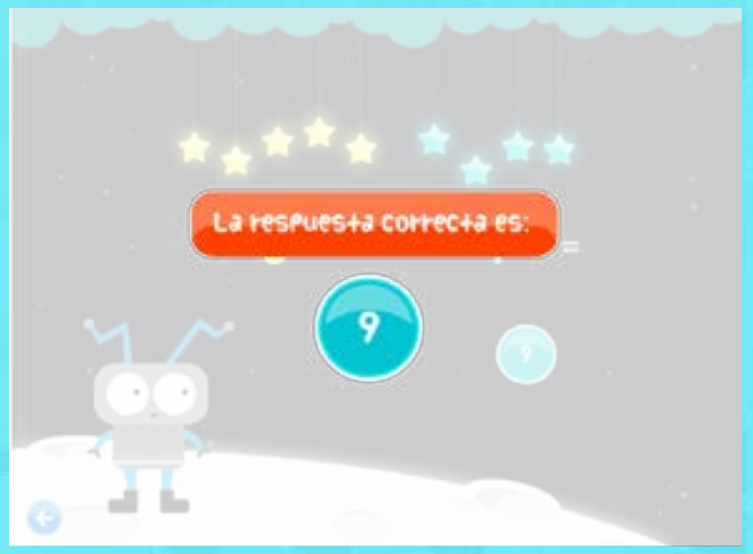

2. grupo de respuestas




respuesta correcta




respuesta incorrecta



# nivel 2

2.2. “jugar”

**¿Cómo jugar?**

El juego consiste en explotar parejas de burbujas. Una burbuja corresponde a una suma y la otra a su respuesta.

barra de tiempo


pantalla al acabarse el tiempo y perder


pantalla al ganar


menú de pausa

3

# nivel 3

Objetivo. aprender a escribir los números correcta-
mente.

# nivel 3

## 3.1. “escribir”

1. Este nivel inicia con un tutorial donde se muestra la trayectoria que debe recorrer el niño con el dedo para escribir el número (dónde debe empezar y dónde debe terminar).

2. El niño debe escribir los números según el tutorial.

3. Se mostrarán mensajes cuando el número se escriba correctamente o cuando se escriba de forma incorrecta.

4. Se puede pasar de un número a otro mediante flechas de navegación.

1.


cantidad de globos representa el numeral